Norges miljø- og biovitenskapelige universitet

Masteroppgave 2021 30 stp
Fakultet for realfag og teknologi

Klassifisering av bevegelse hos kyr ved bruk av tilbakevendende nevrale nettverk

Movement classification for cattle using recurrent neural networks

Knut-Henning Kofoed
Geomatikk – kart, satelitter og 3D-modellering

Forord

Denne oppgaven konkluderer fem uforglemmelige år ved Norges miljø- og biovitenskapelige universitet, NMBU.

Jeg vil først og fremst takke hovedveilederen min Seyed Hossein Chavoshi fra fakultetet for realfag og teknologi for å ha vært til stor hjelp gjennom semesteret og gitt meg gode faglige råd og innspill.

Jeg vil også takke Nofence for å ha bidratt med datagrunnlaget og gjort det mulig for meg å skrive denne oppgaven.

Det må tildeles med stor sorg at min tilleggs veileder Håvard Tveite sovnet stille inn 30. Mai 2021. Til tross for sykdomstilløpet så har Håvard Tveite vært med på å forme denne oppgaven og vært til stor hjelp underveis. Noe jeg personlig er veldig takknemlig for.

Knut-Henning Kofoed

Vestby, 28.06.2021

Sammendrag

Ved å kunne klare å klassifisere atferder hos kalv på beite så vil man kunne gi bonden en nyttig indikator på dyrets generelle velvære, og muliggjøre en mer bærekraftig husdyrbeitedrift. Målet for denne oppgaven var å studere potensialet tilbakevendende nevrale nettverk hadde for klassifisering av atferd hos kalv med hjelp av kun akselerometerdata. Videoobservasjoner og akselerometerdata fra to kalver ble koblet sammen for å danne datasett til maskinlæring. Gated Recurrent Unit (GRU) og Long Short-Term Memory (LSTM) arkitekturer ble prøvd ut på et utvalg av atferder fra tre dannede datasett med to ulike oppdelinger, som til sammen ga 24 forskjellige modeller. Modellene ble optimalisert med de beste parameterne ut ifra et forhåndsdefinert utvalg. Atferdene som ble satt i fokus for denne studien var hvile, bevegelse, beiting, diing og drøvtygging. Resultatene viste at den mer moderne arkitekturen GRU gjorde det bedre enn LSTM i flesteparten av modellene, men at begge slet med å klassifisere atferden diing. Det blir konkludert med at det trengs mer data eller andre typer sensorer for at tilbakevendende nevrale nettverk nøyaktig skal kunne klassifisere og skille mellom flere atferder. Framtiden vil bringe enda større fokus for bærekraftig husdyrbeitedrift, og siden tingenes internett (IoT) stadig sprer seg og allerede har fått feste i enkelte deler av landbruks-industrien, så danner denne oppgaven et godt grunnlag for videre analyse.

Abstract

By being able to classify behaviors of calves on pasture, one will be able to give the farmer a useful indicator of the animal's general well-being and enable a more sustainable animal husbandry. The aim of this thesis was to study the potential of Recurrent Neural Networks (RNN) for classification of behavior in calves using only accelerometer data. Video observations accelerometer data from two calves were linked to form datasets for machine learning. Gated Recurrent Unit and Long Short-Term Memory architectures were tested on a range of behaviors from three formed datasets with two different splits, which together yielded 24 different models. The models were optimized with the best parameters based on a predefined range. The behaviors that were put in focus for this study were rest, movement, grazing, suckle and chewing. The results showed that the more modern architecture GRU did better than LSTM in most of the models, but that both struggled to classify the behavior suckle. It is concluded that more data or other types of sensors are needed for a Recurrent Neural Network to be able to accurately classify and distinguish between several behaviors. The future will bring even greater focus on sustainable animal husbandry. Internet of things are constantly evolving and has already gained a foothold in some parts of the agricultural industry. This thesis does therefore form a good basis for further study.

Innhold

Figurer

Tabeller

1 Innledning

Beskrivelse av oppgavens problemstilling, avgrensninger og aktualitet. Samt formål og begrensninger. Videre settes dette i kontekst, og relatert tidligere forskning krediteres.

1.1 Problemstilling

Studie av atferd hos dyr blir ofte gjort av forskere for å få en bedre forståelse av interaksjonen med miljøet og den fysiologiske tilstanden til dyret. Økologer og biologer studerer bevegelsesmønstre hos dyr for å forstå hvordan habitat-valg, migrasjon, for-opptak og miljøendringer påvirker atferdene hos dyr. Det kan imidlertid være vanskelig å overvåke atferd, spesielt når det er en større flokk med dyr eller dyrene er adskilt over store avstander. Framveksten av digitale sporingsteknikker og nye sensorteknologier de siste årene har åpnet opp for utallige muligheter innen anskaffelser av romlig-temporale data for gjenstander i bevegelse. Dette har også ført til at man kan overvåke husdyr i skalaer som ikke tidligere var mulig. Framveksten av kraftige datamaskiner, programvarer og læringsmodeller gjør det mulig å analysere disse multi dimensjonale dataene for å hente ut nyttige opplysninger fra dyrebevegelser. [4, 5, 6]

Sensorer som er i stand til å måle individuelle atferder hos dyr har lenge vært ansett som en mulig revolusjonerende teknologi-retning innen husdyrbeitedrift. Overvåking av atferd i sanntid gjør det mulig for bonden å ta mer nøyaktige og aktuelle beslutninger om dyrets tilstand og velferd. Teknologien lindrer altså mange av tids- og kostnads-utfordringene knyttet til ettersyn av husdyrhelsen og resulterer i en mer bærekraftig beitedrift. Et eksempel på dette kan være å i sanntid gi bonden informasjon om individuelle dyrs adferder gjennom en tjeneste på nettet eller i en egen app. Utvikling av egne programvarer kan også gjøre denne overvåkningen automatisk hvor kritiske hendelser som død kan varsles til bonden. Av de forskjellige sensortypene så har akselerometeret økt i popularitet innen ekstern sporing. Framskritt innen sensor teknologi har gjort de så små og lette at de enkelt kan festes til dyr uten at de blir for sjenerende, og utviklingen innen batteriteknologi og solceller har gjort det mulig for sensorene å samle eksterne data over lengre perioder. [7]

Man kan med ulike observasjoner fra sensorer danne en modell som klassifiserer bevegelser basert på innkommende sensor data. Disse bevegelses klassene danner grunnlaget for videre analyse, for eksempel hos kyr og kalver så kan sykdom forutsies ved å observere reduksjoner i for-opptak og sosiale interaksjoner. Det kan også tidlig oppdages om en ku/kalv er i mindre aktivitet og kanskje er skadet, eller stresset og er i unormalt mye bevegelse. Ulike sensorer kan brukes i en analyse av bevegelsesmønstre. GPS-antenner er ofte brukt i situasjoner hvor overvåkning og analyse av bevegelser over større områder skal foretas, og ved avgrensede områder kan kamera overvåkning være et alternativ til dette. Akselerometer og gyroskoper er også gode alternativer, som gjerne kan supplere en GPS-sensor for å få et mer helhetlig bilde av et dyrs adferd. En slik analyse vil kunne skille mellom adferds klassene basert på de ulike temporale egenskapene ved datasettet, eksempler på dette kan være posisjons endringer i GPS-data eller signalstyrker fra et akselerometer. [8, 9]

Tradisjonelle manuelle observasjons-metoder for klassifisering av atferder baserer seg på å observere dyr og notere ned tidspunkt for ulike bevegelser, enten ute i felt eller senere med hjelp av video data. Dette er en arbeidskrevende og tidkrevende prosess, og kan i enkelte tilfeller forstyrre storfeoppførselen siden det ofte er nærværende mennesker som gjør observasjonene. Men med

en stor framvekst innen maskin- og dyp-lærings modeller har nye automatiserte, kontinuerlige og ikke-påtrengende systemer potensialet til å forbedre og støtte ledelsesbeslutninger tidligere gjort av mennesker. Maskin- og dyp-lærings modeller krever hovedsakelig mye data for å gi resultater, antallet observasjoner nødvendig vil naturligvis variere med hvilken modell som skal brukes og hvilke adferder som skal klassifiseres. For å i tillegg kunne gi pålitelige prediksjoner må også en viss nøyaktighet opprettholdes ved datainnsamlingen.

Maskinlæring er en applikasjon av kunstig intelligens som anvender algoritmer på en måte som gjør de i stand til å lære av data, og deretter bruker det de har lært til å ta informative beslutninger. Praktisk sett er dyp-læring bare en kategori av maskinlæring som ikke trenger veiledning ved en unøyaktig prediksjon. Dyp-lærings algoritmer kan alene bestemme om en prediksjon er nøyaktig eller ikke gjennom sitt eget nevrale nettverk. Denne studien tar i bruk dype-nettverk og nærmere bestemt tilbakevendende nevrale nettverk, som har muligheter til å huske tidligere viste tilstander i datasettet.

For maskin- og dyp-lærings modeller er det to forskjellige metoder som ofte brukes for å predikere atferder hos dyr. En av metodene baserer seg på algoritmer som ser på trender i datasettene og danner atferds grupperinger uten menneskelig innblanding. Denne metoden blir kalt for ikke-styrt læring og har fordelen ved å kunne predikere atferder uten å måtte bruke forhånds klassifisere data. I styrt-læring blir atferdene som er ønskelig å oppdage definert på forhånd i form av atferds-klasser. Sensor dataene blir tildelt et eget attributt med en av disse atferds-klassene basert på dyre observasjonene, og det resulterende datasettet blir grunnlaget for trening av maskin- og dyp-lærings modeller. Opprettingen av trenings datasett er en av de største utfordringene i all styrt-læring, og blir i tilfellet av bevegelses-klassifisering gjort ved hjelp av video observasjoner og menneskelig bedømming. Denne studien vil hovedsakelig basere seg på sistnevnte, styrt-læring fra observasjons klassifiserte atferder.

1.2 Formål

Denne studien baserer seg på tre produserte datasett hvor datagrunnlaget kommer fra to forskjellige kalver, hvor datasettene kun inneholder akselerometer data. Målet er å kunne predikere atferder som hvile, bevegelse, beiting, diing og drøvtygging med hjelp av tilbakevendende nevrale nettverks algoritmer.

Tilbakevendende nevrale nettverks modeller brukt i studiet baserer seg på arkitekturene Long Short-Term Memory (LSTM) og Gated Recurrent Unit (GRU). Begge blir prøvd ut separat på de tre produserte datasettene og følgende analyse blir utført:

- Nøyaktigheten til klassifisering av atferdene hvile, bevegelse, beiting og diing.
- Nøyaktigheten til binær klassifisering av atferdene hvile og bevegelse.
- Nøyaktigheten til binær klassifisering av drøvtygging hvor atferdene blir delt inn i tygge/ikke tygge.

Prestasjonene til de beste modellene blir tilslutt sammenliknet med resultatet fra en enkel ikke-styrt lærings-metode (*k-means clustering*) med samme datasett og antall klasser.

1.3 Tidligere forskning

Forsøk som innebærer klassifisering av atferder hos storfe ved bruk av konvolusjonelle- og tilbakevendende nevrale nettverk, er blitt gjennomført tidligere med høy nøyaktighet. Et eksempel på dette er gjort i et forsøk hvor data fra seks okser ble kjørt gjennom et nettverk for å predikere bevegelser som ligging, beiting, drøvtygging, bevegelse og enkelte variasjoner av dette. Forsøket

baserte seg på treghetsmåleenheter (IMU) som inneholder både et akselerometer og et gyroskop som derfor gir mer detaljerte data, men på bekostning av høyere strømforbruk. [4]

I en annen studie ble ulike bevegelser hos storfe brutt ned til binære klassifiserings problemer med hjelp av en-mot-alle metoden, som baserer seg på å sette ulike bevegelser opp mot en samling av alle andre atferder. Her ble det kun brukt en tradisjonell maskinlæringsmodell i form av et beslutnings-tre, og ikke noen dyp-lærings metoder. Artikkelforfatterne anbefaler å inkludere ekstra sensor data i form av GPS signaler som gir muligheter for å forbedre resultatet ytterligere. [10].

En studie som bruker GPS data i en tradisjonell maskinlærings teknikk er tidligere gjort hvor atferder som beiting, drøvtygging, bevegelse og andre små unike bevegelser blir predikert. Rapporten konkluderer med at høy frekvens data fra akselerometer kombinert med GPS gir bedre nøyaktighet i klassifiseringen, men denne nøyaktigheten ofrer batteri-levetiden samt betydelig øker mengden data som må lagres. [11]

To andre studier er også verdt å nevne, hvor atferds-klassifikasjonene ble gjort på sau. Her blir spørsmålet angående størrelsene på tidsvinduene for sekvensene også godt belyst, som er en viktig parameter for tilbakevendende nevrale nettverk. [7, 5]

1.4 Begrensninger

Det er gjort et begrenset antall studier av adferds-klassifisering for storfe som har gitt gode resultater. Ut ifra alle de forskjellige storfe atferdene så er det muligheten til å predikere drøvtygging og diing hos kalv som er viktig for å kunne observere og ha kontroll over helsetilstanden. Det er også ønskelig å samtidig kunne predikere andre bevegelser med samme modell, men det finnes begrensinger for hvor mye informasjon som kan hentes fra kun akselerometer data.

Ved å bruke nye forbedrede maskin- og dyp-lærings algoritmer så kan forskere og ulike bransjer støttes opp for å løse forskjellige aspekter ved klassifisering av atferd hos dyr. Maskinlæringsmodellene og teknikkene framført i denne studien begrenser seg først og fremst til akselerometer data for storfe, men kan enkelt utvikles videre til å løse liknende problemer for andre legemer i bevegelse. Man kan også med litt ekstra arbeid bruke modellene og teknikken sammen med data fra andre sensorer.

2 Teori

Følgende kapittel har som formål å danne det teoretiske grunnlaget nødvendig i denne studien.

2.1 Sporingsteknologier og dyreatferd

Ulike sensorer og sporingsteknologier kan brukes i studiet av dyreatferd, de meste brukte er:

- Akselerometer
- Gyroskop
- Magnetometer
- Global Positioning System (GPS)

En treghetsmåleenhet (Inertial Measurement Unit - IMU) måler legemets spesifikke kraft og vinkelhastighet, og er gjort med hjelp av en kombinasjon av akselerometre og gyroskop. Men enkelte treghetsmåleenheter inkluderer også magnetometre, som muliggjør måling av legemets orientering.

GPS er et satellittnavigasjonssystem som muliggjør nøyaktig tredimensjonal posisjonering. Teknologien gjør det mulig å observere hvor et dyr befinner seg, samt hvor det befinner seg i forhold til et annet om begge har GPS-mottaker. Dette kan for eksempel gi et innblikk i samhandlingen mellom dyr og miljøet rundt ved å inkludere geografisk informasjon av beitet.

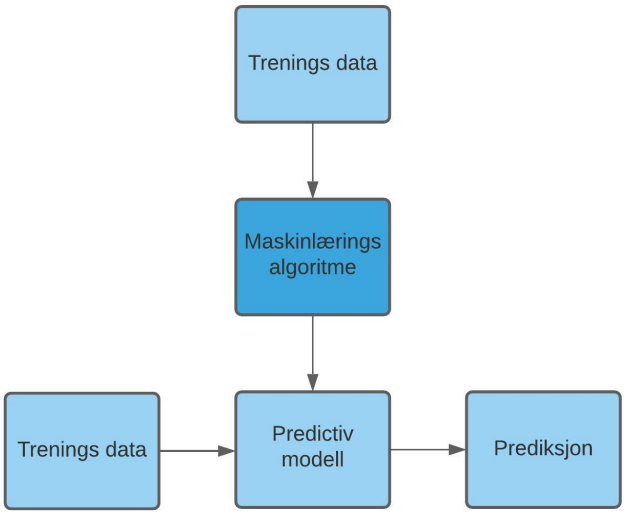
De viktigste atferdene hos husdyr er lik uavhengig om det er snakk om geit, sau eller storfe, noen av disse kan være:

- Hvile
- Bevegelse
- Beiting
- Diing
- Drøvtygging
- Drikking

Mer komplekse atferder kan være spesifikke tilfeller hvor for eksempel dyret parrer seg eller blir jaget av et rovdyr.

2.2 Maskinlæring

På det mest grunnleggende nivået søker maskinlæring å utvikle metoder for datamaskiner til å forbedre ytelsen på visse oppgaver automatisk på grunnlag av observert data. Data er nøkkelingrediensene i alle typer maskinlærings-systemer. Men data uansett mengde er ubrukelige på egenhånd til man får trekket ut kunnskap eller slutninger fra dem [12]. Maskinlæring kan derfor beskrives som en metode for å utvikle en lærings-algoritme basert på en optimal kobling mellom data-domenet og kunnskaps-settet. Basis funksjonene for en maskinlærings-algoritme er vist i figur 2.1. Det finnes mange ulike maskinlærings algoritmer, fordelt på tre ulike kategorier: styrt-, ikke-styrt- og forsterket læring.



Figur 2.1: Flytskjema for implementering av maskinlærings algoritmer.

I styrt læring har man som hovedmål å trene opp en modell med utgangspunkt i allerede kjente data, som lar oss kunne predikere nye ukjente eller fremtidige data [1]. Generering av trenings data er den delen av pre-prosesseringen innen maskinlæring, og er en av de mest utfordrende og tidkrevende delene av slike prosjekt [13]. For dette prosjektet så er enhvert tidspunkt i dataforløpet merket med en aktivitetskategori, som er basert på observasjoner i video opptak.

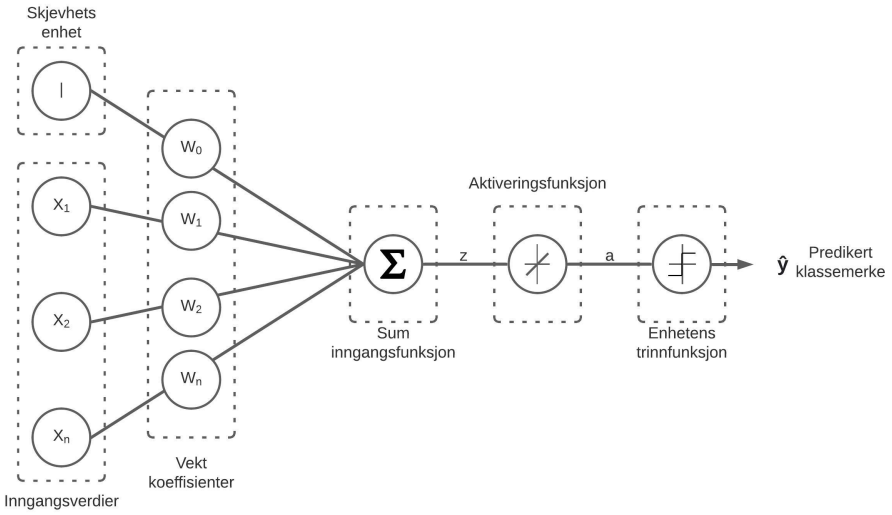
Det er visse situasjoner hvor kjente data er utilgjengelige eller har ukjent struktur. Ved å bruke ikke-styrt læring er det mulighet til å utforske strukturen til dataene våre for å hente ut meningsfylt informasjon og mønster uten noen form for veiledning av kjente data og resultater. I denne oppgaven kan ikke-styrt læring gi en fordel ved at man ikke trenger video observasjoner for å genere trenings og test data.

Målet med forsterket læring er å utvikle et system som forbedrer prestasjonen basert på interaksjoner med dynamiske miljø. Tilbakemeldingen på prestasjonen er gitt i form av en belønningsfunksjon og systemet lærer hvilke handlinger som maksimerer denne funksjonen ved en prøve-og-feil tilnærming. Forsterket læring er ikke mulig å benytte i denne oppgaven og derfor heller ikke en del av teorien.

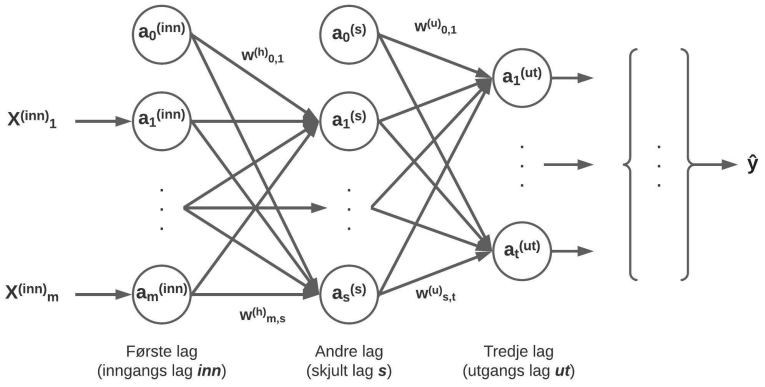
2.3 Nevrale nettverk

Nevroner er de fundamentale bygge blokkene i alle typer nevrale nettverk. Maskinlæring og nevrale nettverk ble bygget på hypoteser og modeller av hvordan den menneskelige hjerne fungerer for å løse komplekse problemer, og har uten tvil fått mye popularitet de siste årene [1]. Figur 2.2 viser

et eksempel på en slikt enkelt nettverk. Med dagens datamaskin kraft er det mulig å koble flere av disse nevrale nettverkene sammen til nevrale nettverk med flere lag. Et fler-lags nevralt nettverk med mer enn et skjult lag blir også kalt et dypt nevralt nettverk. Dype nevrale nettverk kan derfor enkelt sett sees på som nevrale nettverk bare med flere skjulte lag, hvor hvert lag anvender en ikke-linær transformasjon fra inndata til utdata [13]. Figur 2.3 viser strukturen på et dypt nevralt nettverk.



Figur 2.2: Strukturen til en adaptiv lineær nevron algoritme. Inspirasjon hentet fra Sebastian Raschka [1].



Figur 2.3: Strukturen til et multilags nevralt nettverk. Inspirasjon hentet fra Sebastian Raschka [1].

Et fler-lags nevralt nettverk består av to deler, framover-propagering og tilbake-propagering. I framover-propagering propageres dataene fra inngangslaget til utgangslaget og egenskapene til inngangs-dataene blir lært. Deretter beregnes verdiene for utgangslaget som blir sammenliknet med de kjente korrekte verdiene for å kunne evaluere feilen. Minimering av denne feilen gjøres med hjelp av tilbake-propagering. Tilbake-propagering gjør vektjusteringer ved å finne derivater for hver vekt i nettverket og oppdaterer modellen, og prosedyren gjentas gjennom flere epoker for å danne den beste predikeringen.

2.3.1 Konvolusjonelle nevrale nettverk

Konvolusjonelle nevrale nettverk er en dyp-lærings algoritme som gjør det mulig å hente ut spesifikke objekter i data ved å konstruere et såkalt egenskaps-hiarki. Egenskaps-hiarkiet kombinerer flere enkle egenskaper hos datasettet til en mer avansert egenskap ved dataene [1]. Konvolusjonelle nettverk blir for det meste brukt i bildegjenkjenning og talegjenkjenning, men kan også brukes med andre typer data [14], som vist i denne oppgaven.

Konvolusjonelle nettverk består vanligvis av konvulsjons- og samle-operasjoner for å genere dype egenskaper fra rå-dataene. Akselerometer data kan brukes i nettverket for identifiserer de relevante mønstrene til signalene, og de grunnleggende bevegelsene i en aktivitet kan karakteriseres [14]. Ved å legge til flere lag er det mulig å karakterisere mer avanserte bevegelser fra dataene.

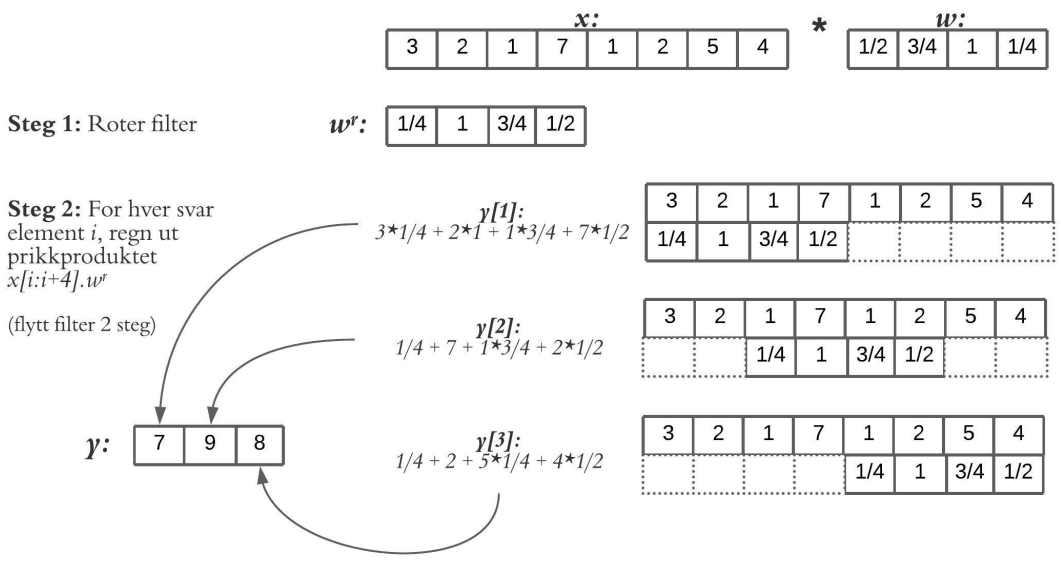
Konvulsjons-lag

Det første laget i nettverket er et konvulsjons-lag som henter egenskaper fra inn-dataene. I denne studien så er dataene representer i en dimensjon, derfor fokuseres det kun på 1-D konvulsjons lag. I de fleste tilfeller bruker man konvolusjonelle nevrale nettverk til bilde-gjenkjennings oppgaver, og da bruker man et 2-D konvulsjons-lag som også har et filter som har to dimensjoner.

Et filter med predefinert størrelse passerer over verdiene i dataene som vist i figur 2.4, og for hvert steg i prosessen blir verdiene i filteret multiplisert med de originale verdiene i dataene filteret overlapper. Denne multiplikasjonen er oppsummert i følgende ligning

$$y = x * w \rightarrow y[i] = \sum_{k=-\infty}^{\infty} x[i-k]w[k]\,, \tag{2.1}$$

hvor x er inn-dataene, w er filteret og y er de nye verdiene.



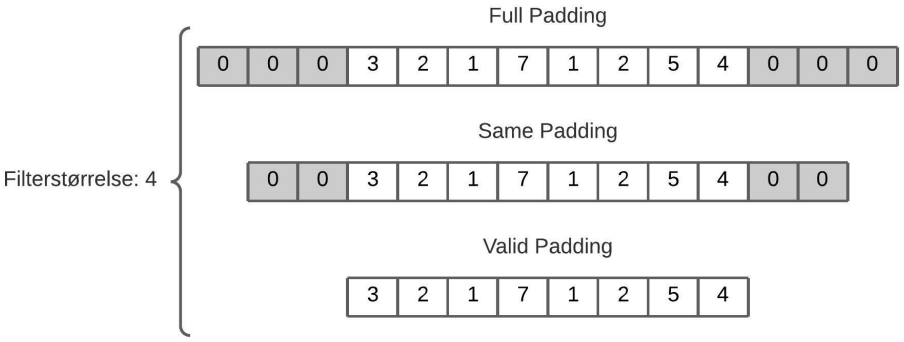
Figur 2.4: Eksempel på en konvolusjons prosess i en dimensjon. Inspirasjon hentet fra Sebastian Raschka [1].

Padding

Padding bestemmer hvordan starten og slutten på datasettet behandles ved konvulsjonen, som vil si hvordan filteret oppfører seg når det passerer over starten og slutten av datasettet. Ved å legge til null-verdier før og etter datasekvensen gir man filteret større plass til å behandle verdiene på endene. Metoden vil gjøre det mulig for inngangslaget å få samme lengde som utgangslaget etter konvulsjons-prosessen. Paddingene mest brukt er *full*, *same* og *valid*:

- *Full padding*: $p = m - 1$
- *Same padding*: $p = m/2$
- *Valid padding*: $p = 0$

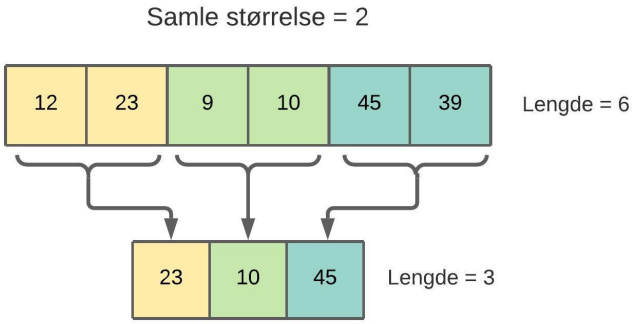
Antall null-verdier som skal legges til for hver ende er gitt av parameteren p, og m er filterstørrelsen. Figur 2.5 viser eksempler for alle disse paddingene.



Figur 2.5: Eksempeler for ulike paddinger basert på filterstørrelse $m = 4$.

Samle lag

Formålet ved en samle-operasjon er å kunne redusere dimensjonaliteten av dataene, men samtidig kunne beholde de viktigste egenskapene. Jo større samle-operasjonen er jo raskere blir algoritmen, men med større informasjons-tap. De mest brukte samle-operasjonene er maks-samling og gjennomsnitts-samling [15]. Det er viktig å få med seg at samle lag ikke har parametere som kan trenes, de har altså ingen vekter eller skjevheter slik som konvolusjonelle lag har. Samle lag reduserer kun størrelsen på inn-dataene for å kunne oppnå en høyere beregningseffektivitet og har derfor ikke noen vekter som kan trenes. Figur 2.6 viser hvordan en 1-D samle-operasjon kan redusere dimensjonaliteten av dataene.



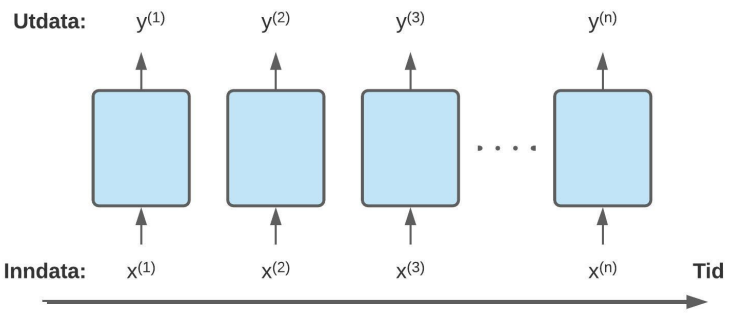
Figur 2.6: Visuell representasjon av et maks-samlings lag i et konvolusjons nettverk, øverste sekvensen representerer data før en samleoperasjon og nedre sekvens etter.

2.3.2 Tilbakevendende nevrale nettverk

Tilbakevendende nevrale nettverk er et dynamisk system som effektivt utnytter tidsinformasjon fra inn-dataene for klassifisering og prediksjon. Så etter at nettverket er trent så vil forholdet mellom ny data og de interne vektene bli behandlet for å produsere et resultat som er basert på tidligere informasjon fra de allerede trente vektene. For slike nettverk så er inngangs-data vanligvis representert som en tidsserie og resultatet som en datasekvens med konstante klasser eller en annen tidsserie. [16]

Sekvenser

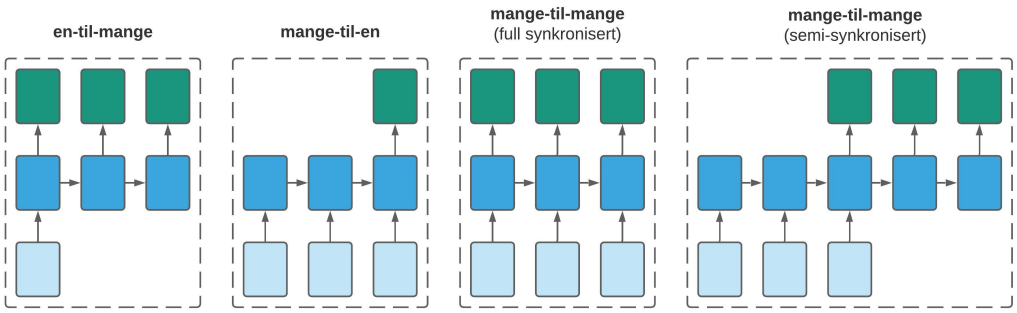
Ved typiske maskinlærings-algoritmer for styrt læring antar man at inngangsdata er uavhengig av hverandre og kan være tilfeldig distribuert. Det som gjør tilbakevendende nevrale nettverk og datasekvenser unike i forhold til andre modeller, er at elementene i en sekvens er avhengig av hverandre og blir representert i en bestemt rekkefølge. I tidsserie data så vil både inndata og utdata naturlig følge rekkefølgen i henhold til tidsaksen som vist i figur 2.7, og derfor er både inndata og utdata betraktet som en sekvens. Siden tilbakevendende nettverk er laget for sekvenser så er algoritmen i stand til å huske tidligere informasjon og behandle nye hendelser deretter.



Figur 2.7: Visuell representasjon av tidsserie data i tilbakevendende nevrale nettverk. Inspirasjon hentet fra Sebastian Raschka [1].

Sekvens modellering har mange bruksområder, fra språk oversettelser til tale gjenkjenning. Derfor må man studere dataene på forhånd, og se på hvilke type relasjon som finner sted mellom inndata og utdata. Figur 2.8 illustrerer de ulike relasjonene man kan støte på når man jobber med sekvens

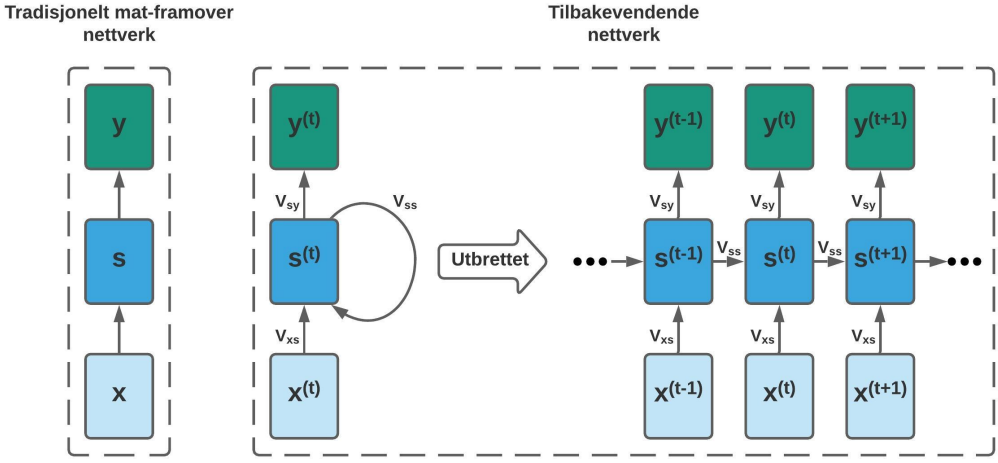
data, enten sekvensene forekommer i inn-dataene, ut-dataene eller begge deler. I tilfellet for denne oppgaven med klassifisering av tidsserier og akselerometerdata blir det brukt mange-til-en modell. En slik modell vil ha sekvenser eller segmenteringer som inndata og en klasse prediksjon for hver sekvens som utdata. Med andre ord så er inndata en rekke vektorer (sekvenser), og utdata en vektor med predikasjonene for alle sekvensene.



Figur 2.8: Data relasjons modeller brukt i tilbakevendende nevrale nettverk, hver boks representerer en vektor. Inspirasjon hentet fra Sebastian Raschka [1].

Struktur og aktivasjoner

Tradisjonelle nevrale nettverk har ikke kompleksiteten til å håndtere variable lengder på inndata sekvenser. Tilbakevendende nevrale nettverk bygger videre på mat-framover modellen og løser problemet ved å ha et skjult lag som henter informasjon både fra inndata laget og det skjulte laget fra tidligere tidstrinnet [17]. Egenskapen ved at modellen kan hente informasjon fra tilstøtende tidstrinn tillater nettverket å ha et minne om tidligere hendelser. Figur 2.9 viser forskjellen på informasjonsflyten i de to modellene, og for å gjøre det mer oversiktlig så er inndata laget x, skjulte laget s og utdata laget y representert som vektorer med flere verdier (sekvenser).



Figur 2.9: Informasjonsflyt for mat-framover og tilbakevendende nevrale nettverk.

Man kan representere den skjulte aktiveringen $s^{(t)}$ som

$$s^{(t)} = f(s^{(t-1)}, x^{(t)}), \tag{2.2}$$

hvor f er en ikke-lineær avbildning. Siden den skjulte aktiveringen $s^{(t)}$ er avhengig av resultatet

fra forrige tidstrinn $s^{(t-1)}$, så må aktiveringene for $t = 0$ være satt til null eller små tilfeldige verdier. Netto inndata beregnes ved å kalkulere den lineære kombinasjonen mellom $s^{(t-1)}$ og $x^{(t)}$ med de tilhørende vektene, og skjevhets vektoren ϕ_s adderes med. Ved å bruke en hyperbolsk tangentfunksjon **tanh** som aktiveringsfunksjon kan man definere aktiveringen av det skjulte laget som

$$s^{(t)} = tanh(v_{ss}s^{(t-1)} + v_{xs}x^{(t)} + \phi_s)\,. \tag{2.3}$$

Utdata laget $y^{(t)}$ er definert som aktiveringen fra det skjulte laget $s^{(t)}$ multiplisert med de tilhørende vektene $v_{(sy)}$, og skjevhets vektoren ϕ_y er lagt til. Ved å bruke en normalisert eksponentiell funksjon **softmax** som aktiveringsfunksjon defineres aktiveringen av utdata laget som

$$y^{(t)} = softmax(v_{sy}s^{(t)} + \phi_y)\,. \tag{2.4}$$

Modellen beskrevet ovenfor tar kun for seg et skjult lag, men kan lett utvides til et dypt nettverk med flere lag.

Gradienter og vekter

Tilbake-propagering gjennom tid blir brukt for å beregne gradienten til taps-funksjonen, og blir brukt når man skal trene vektene til modellen for neste tids epoke. Så for hvert tids-segment så må man summere alle de tidligere bidrag fram til det gjeldende segmentet.

Den grunnleggende ideen bak tilbake-propagering gjennom tid er at det totale tapet L er summen av alle taps-funksjonene fra tiden $t = 1$ til $t = T$ gitt ved funksjonen

$$L = \sum_{t=1}^{T} L^{(t)}\,. \tag{2.5}$$

Tapet ved tidspunkt t er avhengig av de skjulte enhetene for alle tidligere tidspunkt, derfor kalkuleres gradientene med funksjonen

$$\frac{\partial L^{(t)}}{\partial v_{ss}} = \frac{\partial L^{(t)}}{\partial y^{(t)}} \times \frac{\partial y^{(t)}}{\partial s^{(t)}} \times \left(\sum_{k=1}^{t} \frac{\partial s^{(t)}}{\partial s^{(k)}} \times \frac{\partial s^{(k)}}{\partial v_{ss}} \right)\,. \tag{2.6}$$

Leddet $\frac{\partial s^{(t)}}{\partial s^{(k)}}$ er beregnet som en multiplisering av tilstøtende tidstrinn med funksjonen

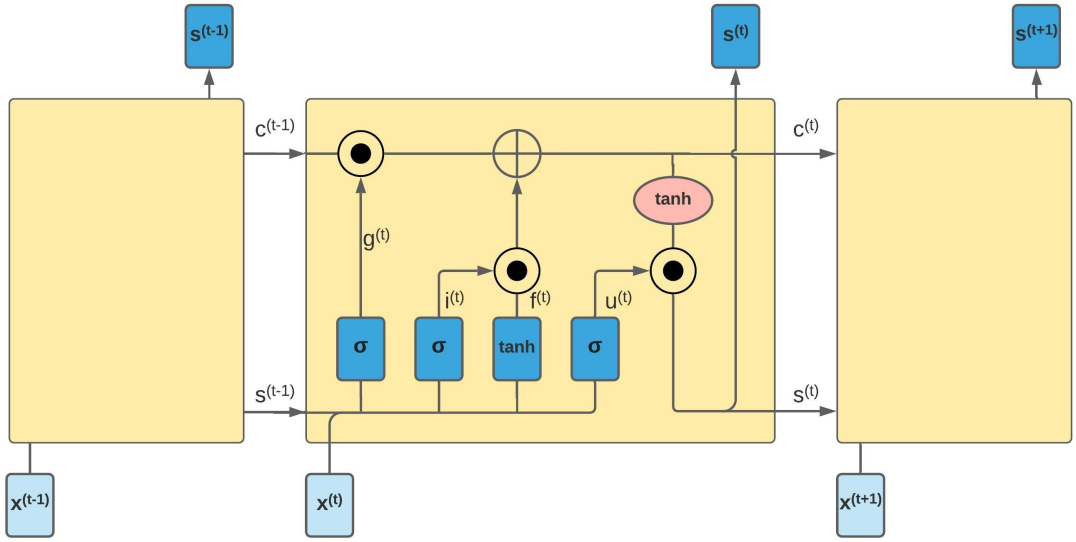
$$\frac{\partial s^{(t)}}{\partial s^{(k)}} = \prod_{i=k+1}^{t} \frac{\partial s^{(i)}}{\partial s^{(i-1)}}\,. \tag{2.7}$$

Siden denne multiplikasjonsfaktoren har $t-k$ multiplikasjoner så vil man da også måtte multiplisere vektene like mange ganger. Problemet med forsvinnende eller eksploderende gradienter oppstår på grunn av disse repeterende multiplikasjonene [1].

Forsvinnende og eksploderende gradienter er et kjent problem med tradisjonelle tilbakevendende nevrale nettverk. Siden slike modeller tilbake-propagerer over lange sekvenser som inneholder små verdier i matrisemultiplikasjonene, så vil gradient verdien krympe for hver tidsepoke og til slutt forsvinne helt. På den måten vil ikke sekvenser lengre unna gjeldende tids-sekvens kunne bidra til parameternes gradientberegning, og vil da ikke kunne lære avhengigheter i data over lengre tidsintervall. Og for eksploderende gradienter så vil gradienten gå mot det uendelige eksponentielt fort og resultere i en ustabil prosess [18]. Long Short-Term Memory (LSTM) og senere Gated Recurrent Unit (GRU) ble introdusert for å løse dette problemet, som blir beskrevet i to neste delkapitlene.

2.3.3 Long Short-Term Memory (LSTM)

LSTM modeller løser problemet med forsvinnende gradienter ved å introdusere minne celler som representerer det skjulte laget. Som beskrevet i forrige delkapittel har et standard tilbakevendende nevralt nettverk kun en aktiveringsfunksjon, i LSTM celler er det fire. Figur 2.10 viser hvordan disse fire interne blå aktiveringsfunksjonene samhandler med hverandre.



Figur 2.10: Long Short-Term Memory celle arkitektur. Inspirasjon hentet fra Olah [2].

Hver linje i figur 2.10 representerer en vektor, linjer som sammenslås betegner sammenføyninger av vektorer og når linjene splittes blir det dannet kopier. Symbolet $\odot$ refereres til element-vis produkt og $\bigoplus$ betyr element-vis summering. Celle tilstanden c flyter langs hele den øverste horisontale linjen i diagrammet, og det er enkelt for informasjon og forflytte seg langs den uendret gjennom flere sekvenser. Den spesielle LSTM strukturen gjør det mulig å legge til og fjerne informasjon fra denne celle tilstanden, dette blir styrt av tre forskjellige porter. Portene er sammensatt av et sigmoid nevralt nettverks-lag σ som gir enten 0 eller 1, og et element-vis produkt $\odot$.

Første steg i en LSTM er å bestemme hva som skal forkastes fra celle tilstanden, dette styres av glemme-porten $g^{(t)}$. Porten tar inn den skjulte enheten fra forrige sekvens $s^{(t-1)}$ og inn-dataene fra gjeldende sekvens $x^{(t)}$, og dette gir et tall mellom 0 og 1 for hvert verdi i sekvensen til celle tilstanden $c^{(t-1)}$. Tallet 1 tilsier at verdien skal beholdes og ved 0 forkastes den fra celle tilstanden. Likningen som beskriver glemme-porten er gitt som

$$g^{(t)} = \sigma(v_{xg}x^{(t)} + v_{sg}s^{(t-1)} + \phi_g)\,. \tag{2.8}$$

Neste steg er å bestemme hvilke nye verdier man skal lagre i celle tilstanden, og har to deler. En inndata port $i^{(t)}$ som bestemmer hvilke verdier som skal oppdateres, og en hyperbolsk tangentfunksjon som genererer en vektor med de nye kandidatverdiene $f^{(t)}$ som kan legges til celle tilstanden. Disse beregnes som følger:

$$i^{(t)} = \sigma(v_{xi}x^{(t)} + v_{si}s^{(t-1)} + \phi_i) \tag{2.9}$$

$$f^{(t)} = tanh(v_{xf}x^{(t)} + v_{sf}s^{(t-1)} + \phi_f) \tag{2.10}$$

Ved å kombinere resultatene fra likning 2.8 og tidligere celle tilstand med 2.9 og 2.10 får man den

nye celletilstanden $c^{(t)}$, og blir beregnet ved likningen

$$c^{(t)} = g^{(t)} * c^{(t-1)} + i^{(t)} * f^{(t)} . \tag{2.11}$$

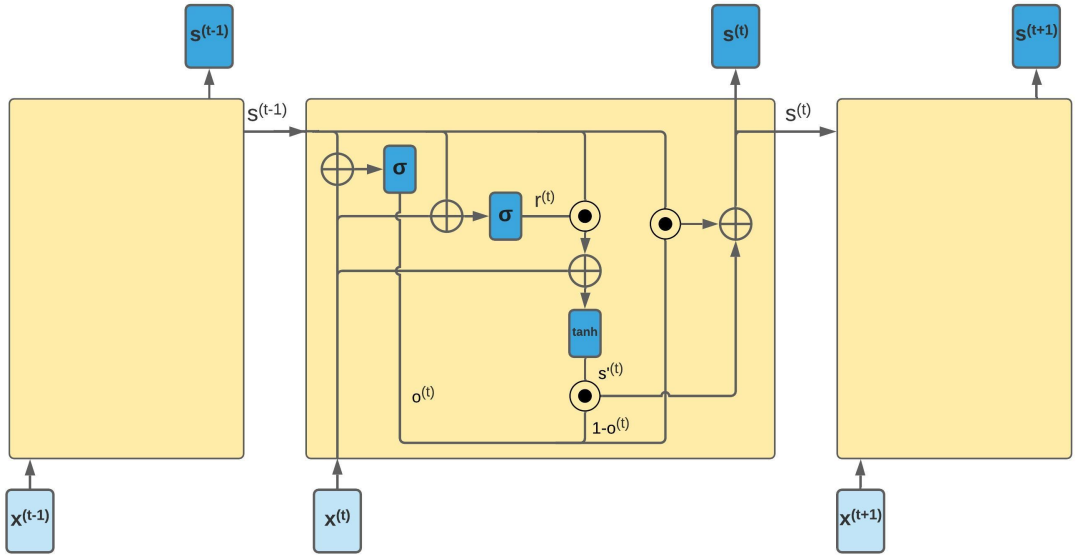
Til slutt må det avgjøres hva som skal leveres videre til den skjulte enheten $s^{(t)}$. Celletilstanden danner utgangs grunnlaget, og blir sendt gjennom en hyperbolsk tangentfunksjon og deretter multiplisert med utgangsporten $u^{(t)}$. Likningen for utgangslaget og den gjeldende skjulte enheten er gitt som:

$$u^{(t)} = \sigma(v_{xu}x^{(t)} + v_{su}s^{(t-1)} + \phi_u) \tag{2.12}$$

$$s^{(t)} = u^{(t)} \odot tanh(c^{(t)}) \tag{2.13}$$

2.3.4 Gated Recurrent Unit (GRU)

GRU er en nyere metode for å løse problemet med forsvinnende gradienter på som gjør det mulig for hver tilbakevendende enhet å adaptivt fange avhengigheter av forskjellig tidsskala. På samme måte som LSTM så har GRU porter som modulerer informasjonsflyten inne i cellen, men uten å ha en separat celle tilstand som minne. Løsningen ligger i såkalte oppdaterings-porter og tilbakestillingsporter som i utgangspunktet er to vektorer som bestemmer hvilken informasjon som skal sendes til utgangslaget. Det unike med portene er at de kan bli opplært til å holde på tidligere informasjon uten å bli påvirket over tid [17]. Figur 2.11 viser hvordan disse portene er brukt sammen med inn-dataene og tidligere informasjon.



Figur 2.11: Informasjonsflyten og strukturen hos Gated Recurrent Unit. Inspirasjon hentet fra Kostadinov [3].

Første steg i GRU er å kalkulere gjeldende verdier for oppdaterings-porten $o^{(t)}$. Dette gjøres ved å kombinere inndata $x^{(t)}$ med informasjon fra forrige tidstrinn $s^{(t-1)}$, og deretter kjøre resultatet gjennom en sigmoid aktiveringsfunksjon σ for å få verdien mellom 0 og 1. Både inn-dataene og informasjon fra forrige tidstrinn er sekvenser og har egne vekter. Oppdaterings-porten hjelper modellen å bestemme hvor mye av tidligere informasjonen som skal gis videre til senere tidstrinn.

Likningen er gitt som

$$o^{(t)} = \sigma(v_{xo}x^{(t)} + v_{so}s^{(t-1)})\,. \tag{2.14}$$

Neste steg er å kalkulere verdier for tilbakestillings-porten $r^{(t)}$, som har i oppgave å bestemme hvor mye av tidligere informasjon som skal glemmes. Likningen er ganske lik som i oppdaterings-porten og er gitt som

$$r^{(t)} = \sigma(v_{xr}x^{(t)} + v_{sr}s^{(t-1)})\,. \tag{2.15}$$

Tilbakestillings-porten vil ha en effekt på de endelige utgangs verdiene ved å introdusere et nytt minneinnhold $s'^{(t)}$ som kan lagre relevant informasjon fra tidligere tidsepoker. Dette minneinnholdet kalles en kandidat-aktivering og er gitt ved likningen

$$s'^{(t)} = tanh(v_{xs'}x^{(t)} + r^{(t)} \odot v_{ss'}s^{(t-1)})\,. \tag{2.16}$$

Til slutt må nettverket beregne den endelige utgangs vektoren $s^{(t)}$ som inneholder informasjon for den gjeldende sekvensen som kan sendes videre. For å få til dette må man ta i bruk resultatet fra oppdaterings-porten som har i oppgave å bestemme hva som skal samles inn fra gjeldende minneinnhold. Likningen for utgangslaget er gitt som

$$s^{(t)} = o^{(t)} \odot s^{(t-1)} + (1 - o^{(t)}) \odot s'^{(t)}\,. \tag{2.17}$$

2.3.5 Aktiveringsfunksjon

Oppgaven til en aktiveringsfunksjon er å bestemme om informasjon skal videreformidles til utgangslaget eller ikke. Et nevralt nettverk beregner den vektede summen av dataene og legger til eventuelle skjevheter og til slutt bestemmer om disse summene skal sendes videre ved bruk av en aktiveringsfunksjon. Hvis et nevralt nettverk ikke er tildelt en aktiveringsfunksjon, så vil nettverket fungere akkurat som en lineær regresjonsmodell hvor predikert data er det samme som inn-dataene. Det er hovedsakelig to typer aktiveringsfunksjoner, lineære og ikke-lineære.

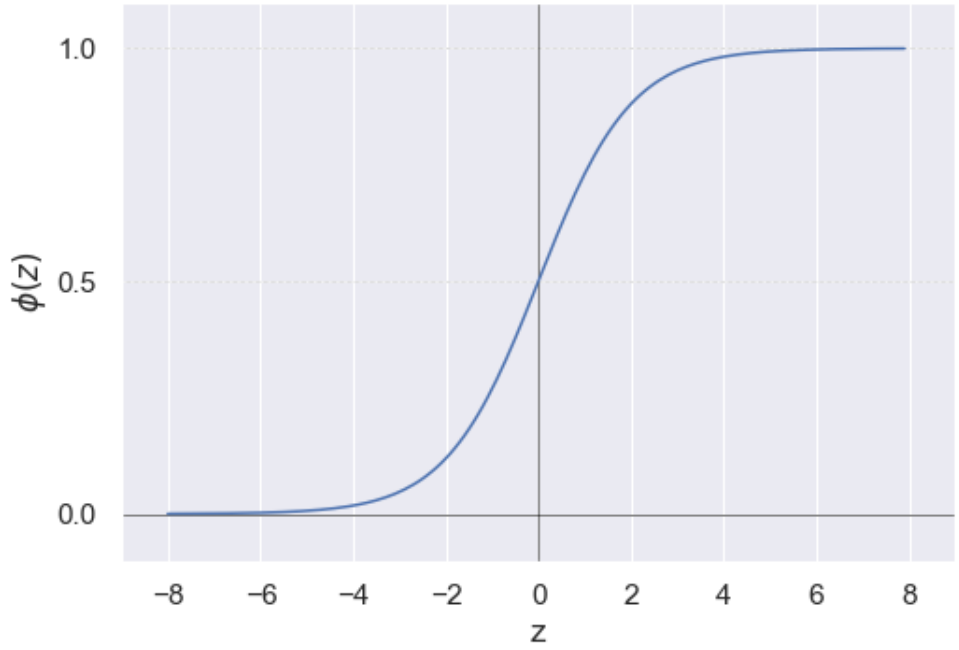
Et nevralt nettverks prediksjons nøyaktighet kan variere ut ifra hvilke type aktiveringsfunksjon som brukes. Lineære aktiveringer har funksjoner som også er lineære, og hvis en slik aktiveringsfunksjon blir brukt så vil nettverket kun ha kapasitet til å tilpasse seg de lineære endringene i inn-dataene. Ikke-lineære aktiveringsfunksjoner foretrekkes framfor lineære fordi de gjenspeiler seg i de ikke-lineære egenskapene i den virkelige verden, og kan gjøre det mulig å lære mer komplekse strukturer i datasettet. [19]

Sigmoid

Sigmoid-funksjonen har en karakteristisk S-formet kurve som vist i figur 2.12 og avbilder netto inndata z på en logistisk fordeling i området 0 til 1. Aktiveringsfunksjonen er gitt som

$$\phi_{logistisk}(z) = \frac{1}{1 + e^{-z}}\,, \tag{2.18}$$

hvor $\phi_{logistisk}(z)$ er sigmoid aktiveringsfunksjonen for netto inndata z. Funksjonen vil kun gi positive verdier videre til neste lag siden den ikke er symmetrisk om origo, dette kan endres ved å skalere sigmoid funksjonen.



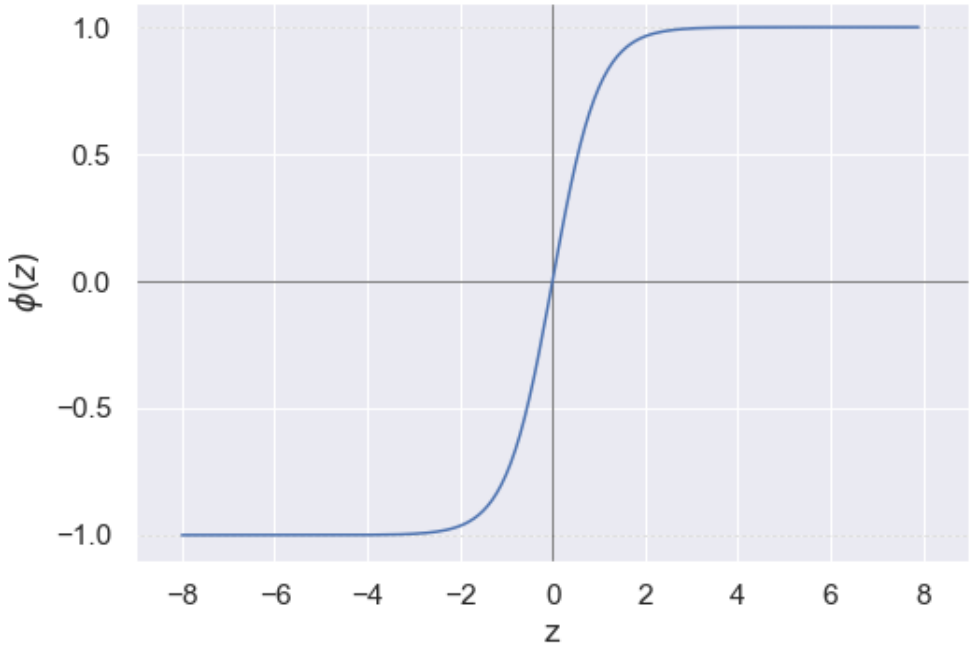
Figur 2.12: Plot for sigmoid aktiveringsfunksjon.

Tanh

Hyperbolic tangent, også kjent som tanh, er skalert versjon av aktiveringsfunksjonen sigmoid. Forskjellen ligger i at tanh funksjonen er symmetrisk om origo og avbilder netto inndata på en fordeling i området -1 og 1, som vist i figur 2.13. Tanh har også en litt brattere gradient. Likningen for tanh funksjonen er gitt som

$$\phi_{tanh}(z) = 2 * \phi_{logistisk}(2z) - 1 = \frac{e^z - e^{-z}}{e^z + e^{-z}}\,, \tag{2.19}$$

hvor $\phi_{logistisk}$ er kalkulert med likning 2.19. Tanh er ofte brukt i de skjulte lagene i nevrale nettverk, og fordelen med tanh er at den har et bredere spekter for utgangs-dataene enn en vanlig sigmoid funksjon.



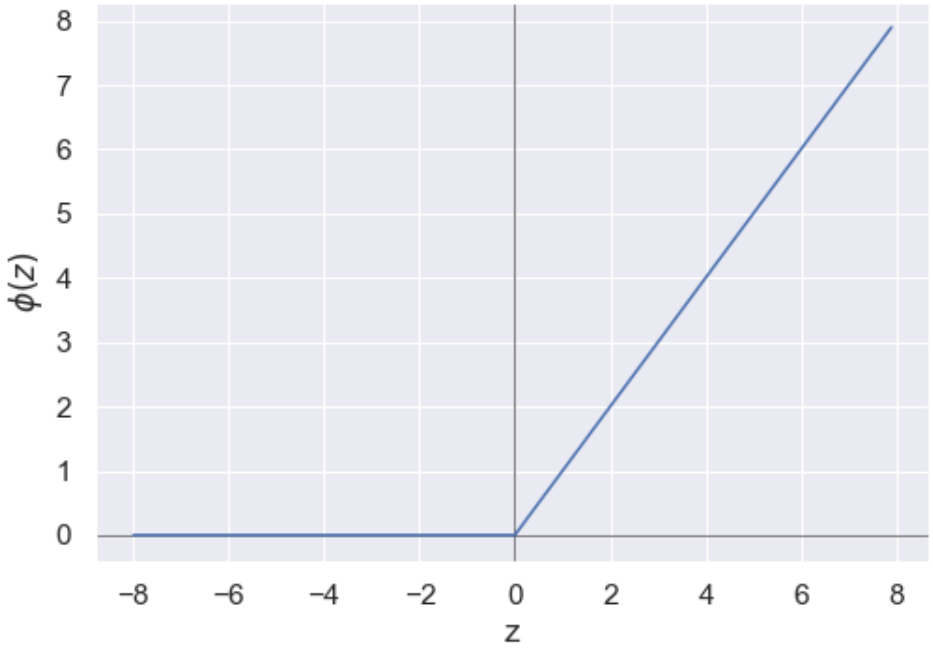
Figur 2.13: Plot for hyperbolic tangent aktiveringsfunksjon.

ReLU

Rectified Linear Unit, også kjent som ReLU, er også en ikke-lineær aktiveringsfunksjon. Likningen for ReLU er gitt som

$$\phi(z) = max(0, z)\,, \tag{2.20}$$

og viser at hvis netto inndata fra likning 2.20 er mindre enn 0, så vil ReLU returnere 0. Figur 2.14 viser grafen for aktiveringsfunksjonen. I de fleste tilfeller så vil ReLU funksjonen yte bedre enn andre aktiveringsfunksjoner i skjulte lag og er mye brukt i nevrale nettverk.



Figur 2.14: Plot for ReLU aktiveringsfunksjon.

Softmax

Aktiveringsfunksjonen softmax blir brukt sammen med utgangslaget til nevrale nettverk og er en kombinasjon av flere sigmoid funksjoner. I sigmoid funksjonene så blir det returnert verdier i intervallet 0 til 1, og disse blir i dette tilfellet sett på som sannsynligheter for at dataene tilhører forskjellige klasser. Funksjonen er gitt som

$$\phi(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} for j = 1, ..., K\,, \tag{2.21}$$

og kan brukes i fler-klasse problemer hvor man har lyst til å se sannsynligheten p for at en netto inngangs verdi z tilhører en bestemt klasse k.

2.3.6 Optimaliserer

Treningen av nevrale nettverk baserer seg på tilbake-propagerings metoden der hvor feilklassifiseringene blir hentet fra utgangslaget, og brukt til å oppdatere vektene tilbake mot inngangs-laget. Disse feilklassifiseringene kan med andre ord være med å bestemme gradienten til taps-funksjonen i forhold til nettverkets vekter, og hovedformålet er å finne verdiene for vektene som gir minst tap. Denne oppgaven ligger hos optimaliserings-algoritmen som bruker gradienten til å justere vektene for å minimere taps-funksjonen.

Adam

Adaptive moment estimator, ofte kalt Adam, er en optimaliseringsalgoritme som bruker momentum og adaptive lærings teknikker til å konvergere nettverket hurtigere. Adam lagrer ikke kun de eksponentielt forfallende gjennomsnittene fra tidligere kvadratiske gradienter v_t i likhet med

Adadelta og RMSprop, men også de ikke-kvadratiske gradientene m_t likt momentum teknikken [20]. Likningene for første og andre momentum er gitt som

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.22}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \,, \tag{2.23}$$

hvor β er forfalls-raten og g er gradienten. Problemet med disse momentumene er at under de første tidsstegene så vil estimatet ha en tendens være partisk mot null. Dette skyldes at vektorene for disse er initialisert med nuller ved første tids-steg. For å kunne motvirke disse skjevhetene så er korreksjonene for første og andre moment gitt som

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.24}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \,. \tag{2.25}$$

For å oppdatere parameterne i Adam blir oppdaterings-regelen

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{2.26}$$

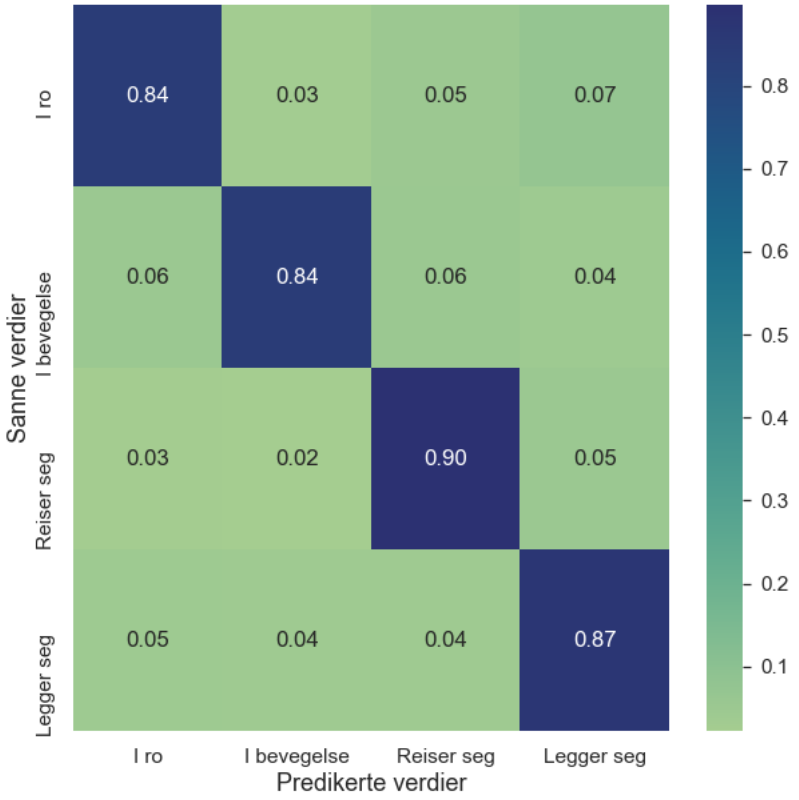
brukt, hvor ϵ er lærings-raten og η trinnstørrelsen.

2.3.7 Nøyaktighetsvurdering

Alle typer maskin- og dyp-lærings algoritme implementeringer må ha en form for evaluering eller mål på ytelse. Det grunnleggende prinsippet er å gjøre en sammenlikning mellom predikasjonene fra algoritmen og de korrekte klassene til dataene, som blir gjort for både trening- og test-dataene. Metoden og framstillingen av ytelse brukt i nøyaktighetsvurderingen vil variere og baserer seg på problemet som skal løses, type data og klassebalansen.

Forvirringsmatrise

Forvirringsmatrisen er en kvadratisk formet matrise som visualiserer nøyaktighetene for hver predikert klasse, og man kan se antall sann positiv, sann negativ, falsk positiv og falsk negativ predikert av maskinlærings modellen. Radene i matrisen representerer de korrekte klassene og kolonnene de predikerte. Verdiene i diagonalen representerer prøvene som ble spådd riktig, og alle verdiene utenfor er feil predikasjoner. Et eksempel på en forvirringsmatrise er gitt i figur 2.15



Figur 2.15: Eksempel på en forvirringsmatrise.

Nøyaktighet

Klassifiserings-nøyaktighet, også referert til som nøyaktighet, er forholdet mellom antallet korrekte prediksjoner og totalt antall predikasjoner. Dette er den mest kraftfulle statistiske metoden for å beregne nøyaktigheten i en modell, men kan fort være misvisende om det finnes ubalanse i klassefordelingen. [1]

Formelen for nøyaktighet er gitt som

$$N = \frac{SP + SN}{FP + FN + SP + SN}, \tag{2.27}$$

hvor SP er sann-positive, SN er sann-negative, FP er falsk-positive og FN er falsk-negative.

Presisjon

Presisjon er et mål for nøyaktigheten av forutsagte positive, også kalt for tillit.

Formelen for presisjon er gitt som

$$P = \frac{SP}{SP + FP}, \tag{2.28}$$

hvor SP er sann-positive og FP er falsk-positive.

Tilbakekalling (Recall)

Tilbakekalling er andelen av sann-positive hendelser som er korrekt predikert som positive, også kjent som sensitivitet. Og er et mål på dekningen av de sann-positive tilfellene.

Formelen for tilbakekalling er gitt som

$$T = \frac{SP}{FN + SP}\,, \tag{2.29}$$

hvor SP er sann-positive, FN er falsk-negative og FP er falsk-positive.

F1-score

Verken presisjon eller tilbakekalling fanger informasjon om hvor godt en modell håndterer negative tilfeller, som vil si antall sann-negative. Ved å kombinere presisjon og tilbakekalling får man F1-score, som er den vektede gjennomsnittet av presisjon og tilbakekalling. Denne måleenheten er et bra nøyaktighets-mål for datasett med ujevn klasseinndeling og tar hensyn til både falske-positiver og falske-negativer. [21]

Formelen for F1-score er gitt som

$$F1 = 2 * \frac{P * T}{P + T}\,, \tag{2.30}$$

hvor P er presisjon og T er tilbakekalling.

Resultatene i denne studien vil bli visualisert med forvirringsmatriser, og F1-score vil være en av hovedmålene brukt i nøyaktighetsvurderingen fordi den tar hensyn til skjev klassefordeling på en måte som gir et mer helhetlig bilde av prestasjonene til modellene. Presisjon og recall vil også bli vist i egne tabeller.

Taps-funksjon

For å få et mål på hvor godt en algoritme lærer av dataene bruker man en taps-funksjon. Funksjonen brukes under trening av nevrale nettverk, og gir en taps-verdi som forteller oss noe om gapet mellom predikerte verdier og korrekte verdier. Størrelsen på disse gapene blir gitt videre til en optimaliserings-funksjon og brukt til å få modellen til å lære bedre og dermed redusere taps-verdien for de kommende epokene. Målet for et nevralt nettverk er derfor å få lavest mulig taps-verdi, men under en modell evaluering så må riktig taps-funksjon velges og det finnes taps-funksjoner for både binære data og kategorisk multi-klassifisering.

Kategorisk kryss-entropi er brukt i eksempler hvor man har flere klasser å klassifisere, og man må som regel vektorisere klassene slik at de består av 0 og 1 verdier. Likningen for en kategorisk taps-verdi er gitt som

$$D_{kategorisk}(\hat{y}, y) = -\sum_{i=1}^{k} y_i log(\hat{y}_i)\,, \tag{2.31}$$

hvor y er korrekt klasse og $\hat{y}$ er predikert klasse.

3 Datasett og metoder

Beskrivelse av datasettet, metoder for å genere trenings-data samt forklaring på ulike typer tilbakevendende nevrale nettverk og modeller.

3.1 Datasett

Datagrunnlaget baserer seg på et forsøk med kallenavnet Kalvelykke som ble startet av Norsøk og Nofence hovedsakelig for å undersøke fordelene ved å la kalven gå på beite med mora over lengre tid, og mulighetene til å oppdage atferden diing ute på beitet med akselerometerdata. Datainnsamlingen ble gjort ved et gårdsbruk i Rennebu kommune, som ligger i Trøndelag fylke. Figur 3.1 viser lokasjonen og flyfoto over beiteområdet, og størrelsen på det inngjerde området er rundt 45 mål. Det ble gjort test datainnsamlinger og ulike observasjoner fra midten av Juli til slutten av August 2020, men dataene fra kalv er hentet inn 23. og 24. August.

Figur 3.1: Geografisk område for forsøket.

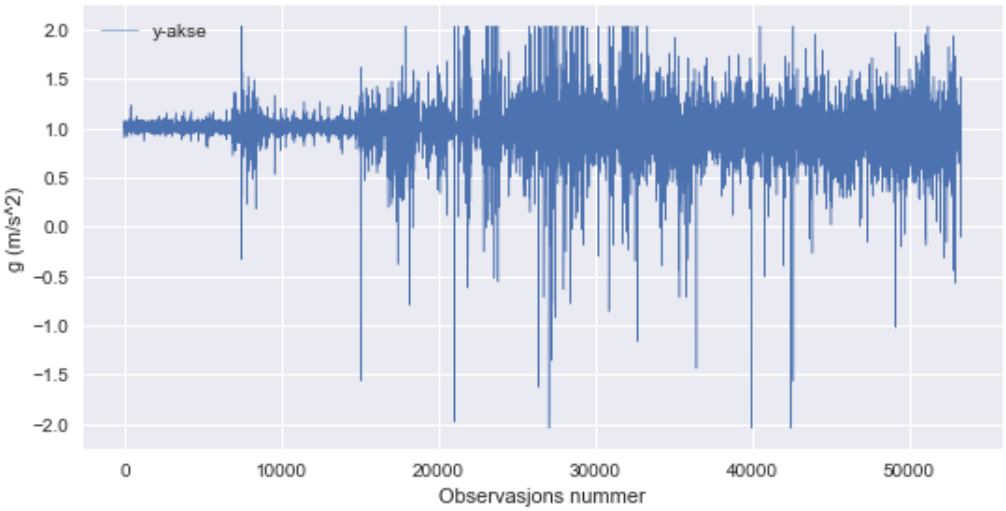
Den 23. August var det overskyet regn og den 24. var det sol og vekslende skydekke. Beiteområdet ligger på en høyde mellom 550 og 560 meter over havet og består for det meste av gress, men også en liten andel lav skog.

3.1.1 Akselerometer

Et akselerometer brukes ofte for å måle hastighets endringer, støt og vibrasjon i en lokal treghetsramme. De fleste akselerometer måler akselerasjon i tre ulike akser og har ulike oppløsninger både i form av oppdaterings-frekvens og følsomhet på målinger. Et akselerometer bruker veldig lite strøm, bare en brøkdel av for eksempel et gyroskop eller GPS med samme oppdaterings-frekvens. Akselerometer brukes ofte i små-elektronikk som mobiler og kameraer for navigering og bildestabilisering, men kan også brukes i en rekke andre ingeniør industrier. Disse egenskapene passer derfor bra som utgangspunkt i bevegelses-klassifisering for dyr og mennesker.

Akselerometer sensorens oppdaterings-frekvens kan ha en stor betydning på prestasjonen til ulike maskinlærings-modeller. For høy oppløsning vil resultere i lange trenings tider for modellen, og i verstefall ikke fungere i det hele tatt på grunn av manglende datamaskin minne. Høye oppdateringsfrekvenser vil også gi høyere strømforbruk, noe man vil unngå i en batteri-klave laget for å være operativ i flere måneder. For lav oppløsning vil resultere i at modellen går glipp av mye informasjon den trenger for å gjøre korrekte klassifiseringer. Lave oppdaterings-frekvenser vil også gjøre det umulig å oppdage bevegelser med høyere frekvens. Valget av oppdaterings-frekvens vil derfor variere for ulike bruksområder. Oppdaterings-frekvensen til sensoren i brukt for denne studien er 10Hz.

Valg av sensorens følsomhet vil også variere med bruksområdet, følsomheten angir hvor mye bevegelse sensoren kan oppdage. For høy følsomhet vil ikke ha noe effekt på maskinlærings-modellen fordi datasettet normaliseres før bruk, noe som fjerner eventuelle ekstreme signalverdier sett på som støy. Og det vil derfor være bortkastet siden de ofte er dyrere i innkjøp og er mer komplekse. For lav følsomhet vil resultere i at dataene ikke får fanget opp høye verdier av akselerasjon, og vil derfor eventuelt gå glipp av viktig informasjon. Sensoren i denne studien har en følsomhet på $\pm 2\,\mathrm{g}$ for hver akse, som vil si at sensoren kan måle signaler på høyst $2\,\mathrm{g}$ og minst $-2\,\mathrm{g}$ for x, y og z aksene. Ved å se på signalkurvene for en av aksene så kan man oppdage at en følsomhet på $\pm 2\,\mathrm{g}$ er for lite. Gravitasjonskraften som virker på enhver legeme på jordoverflaten er $1\,\mathrm{g}$, dette kan observeres i Figur 3.2 ved at signalverdien til y-aksen konstant ligger på rundt $1\,\mathrm{g}$. Problemet med dette er at siden maks signalstyrke akselerometeret kan fange opp er $2\,\mathrm{g}$ og gravitasjonen konstant bruker $1\,\mathrm{g}$ av dette, så vil man ha $1\,\mathrm{g}$ igjen til å fange opp bevegelse. Dette kan observeres i Figur 3.2 ved at man har tydelige avskjæringer på rundt $2\,\mathrm{g}$.



Figur 3.2: Eksempel på akselerometer signal fra y-akse.

Akselerometer dataene inneholder 11 attributter. **serial** er attributten som beskriver hvilken klave observasjonen peker til, og er representert som et unikt heltall. **date** er et datotidspunkt som representerer tidspunktet observasjonen ble motatt på serveren. **header_date** peker til tidspunktet observasjonen ble gjort av akselerometeret, og er derfor brukt til å koble video observasjonen til dataene. Både **date** og **header_date** er oppgitt i tidssonen UTC og blir kun oppdatert hver 32

observasjon, derfor er det laget et eget attributt for å skille disse med navnet **index**. **index** er som et resultat av dette et heltall mellom 0 og 31. **x**, **y** og **z** som er rå verdiene akselerometer sensoren gir ut. **xcal**, **ycal** og **zcal** som er akselerasjons komponentene oppgitt i **g** krefter. Den siste attributten **norm** er beskrevet som kvadratroten av kvadratsummen til komponentene **xcal**, **ycal** og **zcal** som vist i formelen

$$norm = \sqrt{xcal^2 + ycal^2 + zcal^2}\,, \tag{3.1}$$

og blir brukt som et mål for generell bevegelse uavhengig av akseretning.

Akselerometeret er montert inne i en beholder sammen med GPS-antenne, blåtann, batteri, solcellepanel og 2G/4G-mobilt nettverks antenne. Figur 3.3 viser hvordan klaven med beholderen er fester på kalven i forsøket. Dataene fra akselerometeret blir sendt over 2G-mobilt nettverk til Nofence sine servere, og må derfor ha mobildekning for å fungere.

Figur 3.3: Bilde av kalv og klave med beholder for elektronikk.

3.1.2 Video

Video opptak er ofte brukt i forsøks prosjekt for å kunne ha muligheten til å analysere forsøket etter at feltdelen av prosjektet er fullført. I dette tilfellet er video opptak brukt for å kunne ha muligheten til å klassifisere ulike bevegelser, og kunne skape egne datasett for dette basert på behov og ønsker. Klassifiserings datasett skapt fra opptakene er nødvendig for å kunne trene opp maskinlærings modellene, og nøyaktigheten til modellene samsvarer med nøyaktigheten til klassifiseringene. Så gode video data med tilstrekkelig oppløsning er viktig for forsøket.

Video oppløsning er en av de viktigste egenskapene for opptak man må se på under et forsøk. Optimalt valg av oppløsning vil kunne gi de detaljene nødvendig for klassifiseringen og samtidig kreve minst mulig plass på et lagringsmedium. Det er flere forhold man må se på for å kunne velge den perfekte oppløsningen; skal kameraet stå i ro eller hvilke avstander fra observasjons objektet er det snakk om? I dette forsøket er det filmet i full HD, altså 1920x1080, som er en standard på alle moderne kameraer. Denne oppløsningen viser seg å være funksjonell i de fleste tilfeller for dette forsøket.

Bildefrekvens er en annen egenskap som kan være viktig for opptak i et forsøk. Optimalt valg av

bildefrekvens vil gi samme utslag som for video oppløsning nevnt i avsnittet ovenfor. Bildefrekvens er målt som bilder per sekund, og moderne håndholdte kameraer for privat bruk kan filme i frekvenser opp til 250 bilder per sekund eller mer i full HD. I dette forsøket ble det filmet med 30 bilder per sekund, og fungerer utmerket for bevegelses-klassifiseringen som er gjort.

For å kunne koble sammen observasjoner i videoen og akselerometerdata trenger man å vite tidspunktet videoen ble tatt. Så det er sammen med videoen lagt ved en tekstfil som representerer bildetidspunktet til hvert sekund av videoen. For å kunne klare å vite dette er det i en vilkårlig del av klippene vist fram en riktig innstilt klokke. Ved å i tillegg vite bildefrekvensen til videoklippet er det mulig å kalkulere tidspunktene for hele videoen med et sekunds nøyaktighet, som er tilstrekkelig i dette forsøket.

3.2 Programmeringsspråk og verktøy

Tillegg B beskriver framgangsmåten, rekkefølgen og strukturen av kodingen brukt og produsert i denne studien. Programmeringsspråket brukt i denne oppgaven er Python.

3.2.1 Python

Python er et programmeringsspråk som består av flere biblioteker og pakker for lesing, skriving, modifisering og analyse av data. Python er betraktet som lett å lære og forstå, og er et populært verktøy for maskinlæring og dataanalyse [22]. For denne studien blir Python 3 brukt til å behandle akselerometerdata og observasjoner. Innlesing, sammenkobling, visualisering og analyse av dataene oppnås ved å bruke en kombinasjon av ulike biblioteker og moduler sammen med funksjoner produsert spesifikt for studien.

Pandas

Pandas er et åpent dataanalyseverktøy for Python som gir raske og fleksible datastrukturer designet for å gjøre arbeidet med relasjonelle data eller data med etiketter, og er godt egnet for tidsserier, sekvensiell data og matrisedata [23]. I denne studien så blir Pandas brukt til å laste inn CSV filer til Pandas datarammer, som er todimensjonale tabelldata med merkede akser (rader og kolonner). Ved hjelp av Pandas så kan man enkelt gjøre pre-prosessering på disse datarammene i Python for å klargjøre dataene til maskinlæring.

Scikit-Learn

Scikit-Learn er et brukervennlig åpent maskinlærings-bibliotek for Python, og inneholder effektive verktøy for predikativ data-analyse. Scikit-Learn kan brukes til klassifisering, regresjon, gruppering og dimensjons-reduksjon, men brukes i denne studien til pre-prosesserings steg som skalering og normalisering av dataene.

TensorFlow - Keras

Keras er et høynivå applikasjonsprogrammeringsgrensesnitt (Application Programming Interface - API) for å utføre nevrale nettverksoppgaver som er skrevet i Python og kjører på toppen av TensorFlow. Keras har støtte for kunstige-, kovolusjonelle- og tilbakevendende nevrale nettverk, samt kombinasjoner av disse. Fordelen ved Keras er at man kan trene modellene både på prosessoren og grafikkortet [1]. Keras blir i denne studien brukt til å bygge opp og trene de ulike nevrale nettverkene.

3.3 Maskinvarespesifikasjoner

Tilgjengeligheten på datamaskin-kraft er kjent for å være avgjørende under trening av maskinlærings-algoritmer, både når det gjelder gjennomførbarhet og tidsbruk. Maskinvarespesifikasjonene setter begrensninger for kompleksiteten på modellene og størrelsen på datasettet. Å ha tilstrekkelig med dedikert minne - Random Access Memory (RAM) for enten prosessoren eller grafikkortet er viktig for gjennomførbarheten og tidsbruken, dedikert minne er raskt og man vil kunne unngå gjentakene operasjoner mellom en treg disk og prosessoren/grafikkortet om hele datasettet kan lastes inn på dedikert minne. Klokkehastigheten og den generelle styrken på prosessoren/grafikkortet vil derimot kun gi effekt på tidsbruken til algoritmene, og er derfor ikke like viktig som dedikert minne.

Maskinlærings-algoritmene er i denne studien kjørt på et system med følgende spesifikasjoner:

- Grafikkort: GeForce GTX Titan X, 12GB dedikert minne.
- Prosessor: Intel Core i7 4771 med 16GB dedikert minne.

Hvor selve treningen av modellene er gjort på grafikkortet.

3.4 Manuell video observasjons klassifisering

Korrekt klassifisering av data er viktig i et forsøk for å danne datagrunnlag til styrt-læring, men å kunne bedømme hva som er korrekt kan være en utfordring. Menneskelige prosesser kan alltid være en kilde til feil, nøyaktigheten til maskinlærings-modeller er derfor direkte avhengig av prestasjonen til de menneskelige manuelle klassifiseringene. Det er blitt nevnt at det ofte brukes video observasjoner for klassifiseringer av bevegelses-data, og dette blir også brukt i denne studien.

For å kunne notere ned ulike observasjoner av bevegelser og senere koble disse opp mot dataene fra et akselerometer, må tidspunktene i videofilene synkroniseres med tidsobservasjonene fra da videoen ble filmet. Dette er allerede gjort av Nofence i datagrunnlaget for oppgaven i form av en CSV fil, men det er viktig å poengtere at forsøket ble gjort på sommeren og at disse tidspunktene er oppført som Norsk sommertid (UTC+2). Denne forskjellen blir rettet i pre-prosesserings delen.

Den manuelle observasjons klassifiseringen går ut på å danne to tabeller i et regneark program som Excel, en tabell for bevegelses observasjoner og en annen for observasjoner av drøvtygging. Her blir identifikator for kalven, tidspunktet og type aktivitet notert etter hvert som det blir observert noe under gjennomgangen av videoene. Det viktige er å føre dette så korrekt som mulig inn i regnearket med egne rader for når en aktivitet starter og slutter. Tabell 3.1 viser føringseksempel for aktivitets notering.

Tabell 3.1: Eksempel på aktivitets føring i regneark.

Nofence ID	Type dyr	Tid	Aktivitet
35396	Hvit egenkalv	2020-08-23 16:40:15	VIDEO START
35396	Hvit egenkalv	2020-08-23 16:52:03	Bevegelse start
35396	Hvit egenkalv	2020-08-23 16:52:11	Bevegelse slutt
35396	Hvit egenkalv	2020-08-23 16:52:50	VIDEO SLUTT

For å gjenkjenne dyrene i videoene har klavene ulike farger som slås opp i en tabell for finne den unike identifikatoren brukt i akselerometer-dataene. Tabell 3.2 viser de ulike navnene til kuene/kalvene koblet opp mot de unike identifikatorene.

Tabell 3.2: Identifikatorer for storfe.

Type dyr	Nofence ID
Hvit egenkalv	35396
Rosa egenkalv	37368

Bevegelses klassifisering er delt inn i fire aktiviteter; hvile, bevegelse, beiting og diing. Alle aktivitetene sammen med tidspunkt er notert ned når de starter og når de slutter som vist i Tabell 3.1, unntatt hviler. Aktiviteten hviler blir sett på som de tidsintervallene når dyret slutter med en aktivitet og fram til en annen starter, og blir senere automatisk gjenkjent av en funksjon i Python. For den binære klassifiseringen av drøvtygging er kun delt inn i to aktiviteter; tygging og ikke tygging. Notert som tygge start og tygge slutt i regnearket.

Det kan også observeres i Tabell 3.1 at det blir notert når en video starter og når den slutter. Dette er gjort for at et program senere skal kunne hente ut de aktuelle tidsintervallene, som resulterer i at bare relevant akselerometerdata blir hentet inn i dataminnet.

3.5 Preprossesering

For å lettere kunne pre-prosessere data er det i denne studien blitt bearbeidet en pre-prosesserings programpakke i Python tilpasset oppgavene som skal løses.

3.5.1 Aktivitets observasjoner

Regnearket med CSV filformat hvor aktivitetene til kalvene er notert blir hentet inn i Python ved hjelp av Pandas programpakken. For å synkronisere tid-sonen til attributten **Tid** (UTC+2 Norsk sommertid) til UTC trekkes det fra nøyaktig to timer fra alle tidspunktene i kolonnen. Den korrigerte aktivitetstabellen mates inn i et program som bruker Pandas til å vise en ny tabell med kun de radene som har aktiviteten video start/slutt, denne tabellen brukes for å vite hvilke tidsintervall med data som skal hentes fra akselerometerdata filene.

3.5.2 Akselerometerdata

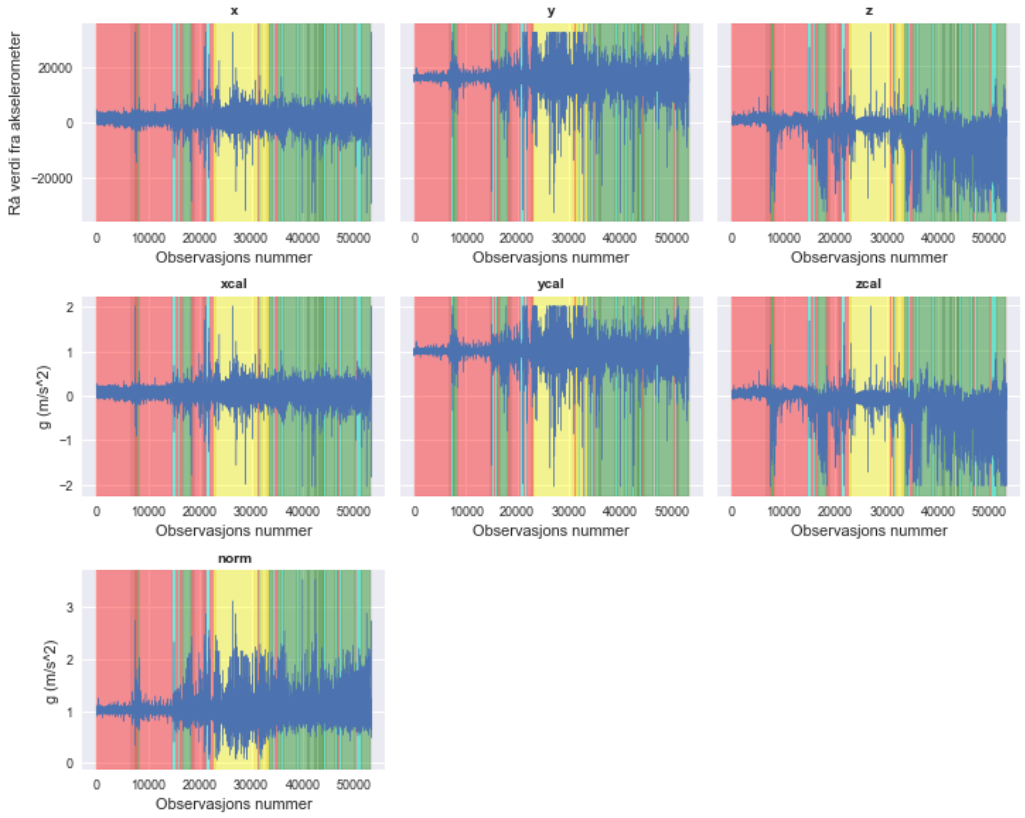
Før akselerometer-dataene hentes inn i Python blir det gjort en spørring på aktivitetsregistreringene som gir ut de unike identifikatorene for kalvene i tabellen. Dette brukes sammen med tidsintervallene for aktivitetsobservasjonene til å finne de relevante datalinjene i akselerometerdataene. For importering av akselerometerdata blir en egen funksjon brukt som bygger på Pandas programpakken. Siden akselerometer-dataene for hvert dyr er representert med en egen fil så kjøres det en loop som henter ut de filene representert av de unike identifikatorene fra aktivitetsregistreringene. Funksjonene bruker deretter en spørring på filen som henter ut de relevante tidsintervallene med observasjoner, slik at kun hentes akselerometerdata i tidsrommet aktivitetsregistreringene ble gjort. Tids attributtene **date** og **date** i akselerometerdataene er kun oppdatert hver 32. observasjon og kan observeres ved at attributten **index** går fra 0 til 31 før tiden oppdateres. For å øke tids oppløsningen på sensor observasjonene så kjøres de gjennom en funksjon som gir hver observasjon en tilnærmet korrekt tid, man kan si at datasettet blir glattet ut.

3.5.3 Sammenkobling - danne datasett

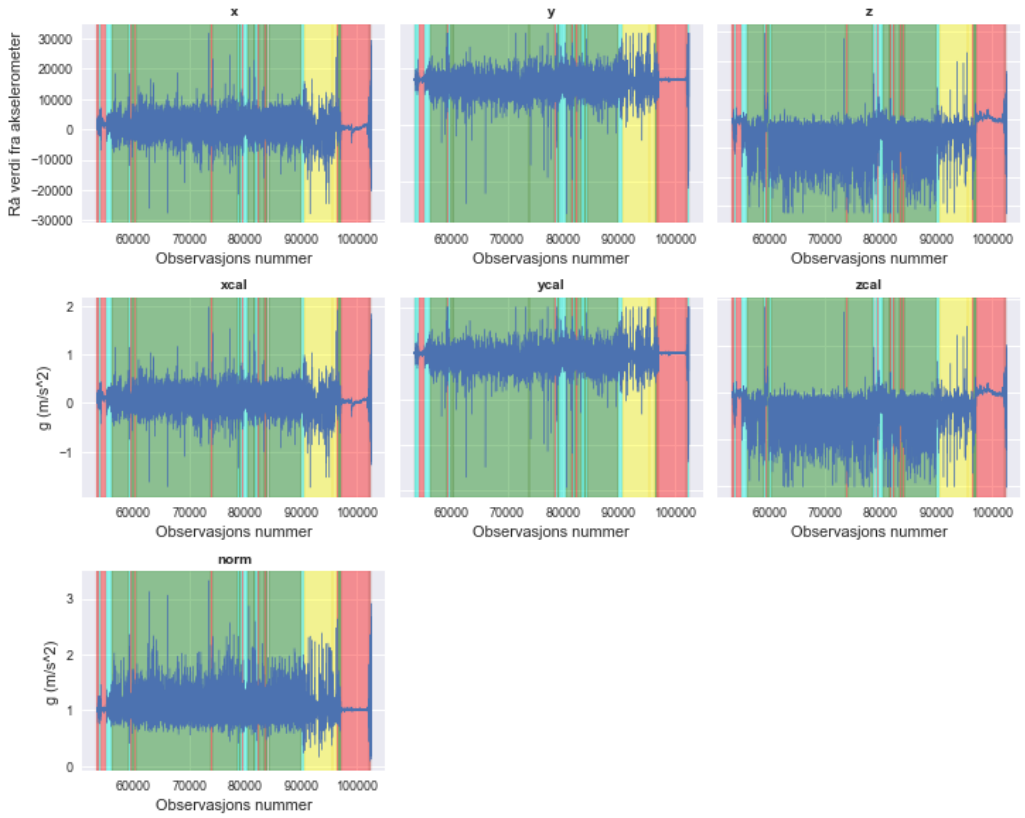
Datasett for maskinlæring trenger dataobservasjoner med tilhørende klassifiseringer, derfor må aktivitets-observasjonene kobles sammen med akselerometer-dataene. Disse to tabellene kjøres gjennom en funksjon som gir ut akselerometer-dataene med en ekstra attributt **aktivitet** hvor

aktiviteten er representert av et nummer. For å få til dette itereres det gjennom aktivitets-observasjonene og det blir for hver iterasjon dannet en maske som representerer de akselerometer-observasjonene hvor en aktivitet skal defineres. En rekke IF tester i funksjonen passer på at gjeldende aktivitet i iterasjonen blir gitt til alle radene i den utvalgte masken. Sluttresultatet blir en tabell med klassifisert akselerometerdata klar til bruk i maskinlærings-applikasjoner.

Figurene 3.4 og 3.5 representerer informasjonen i de endelige datasettene hvor de ulike grafene viser forskjellige attributter. Videre i maskinlæringen brukes kun attributtene **xcal**, **ycal**, **zcal** og **norm**. De forskjellige fargede områdene representerer klassen signalene tilhører hvor fargen rød er hvile, lys blå er bevegelse, grønn er beiting og gul er diing. Grafene brukes til å oppdage feil i korrelasjonen mellom akselerometer signal og tildelt klasse. Eksempelvis kan man i dette tilfellet observere at klassen hvile (rød) har små svingninger i akselerometer signal, som er forventet.

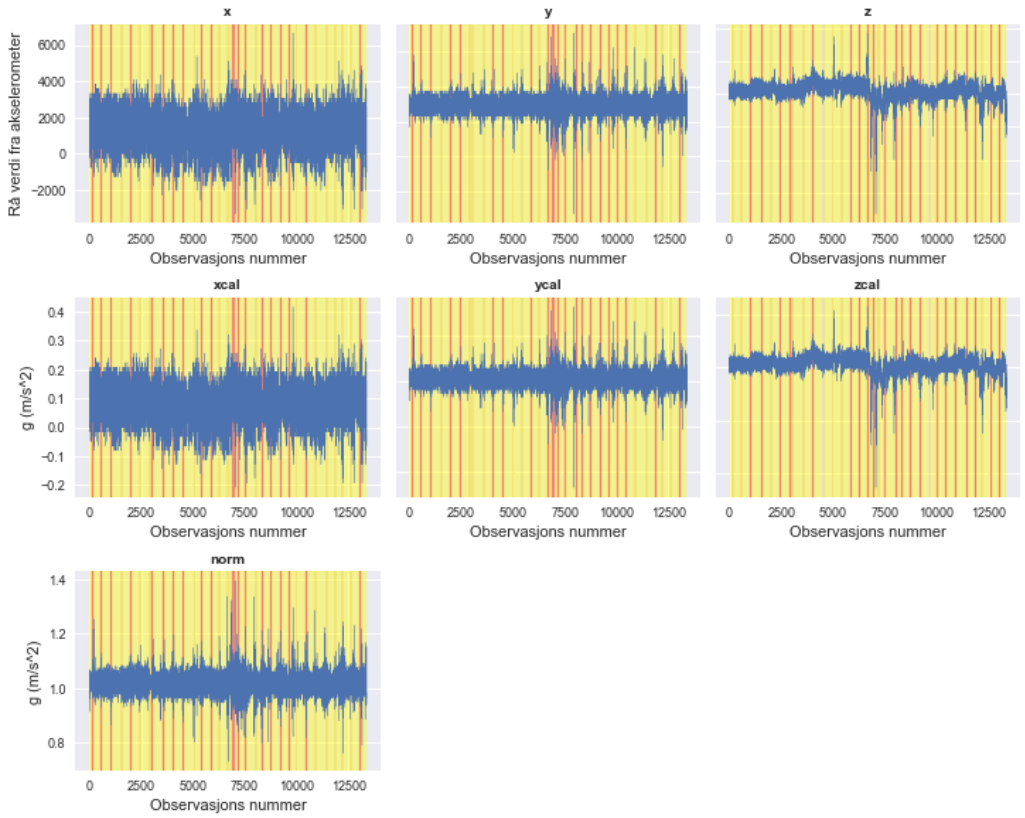


Figur 3.4: Akselerometer signal visualisert for fler-klasset bevegelse, med klave ID 35396 (Hvit egenkalv).



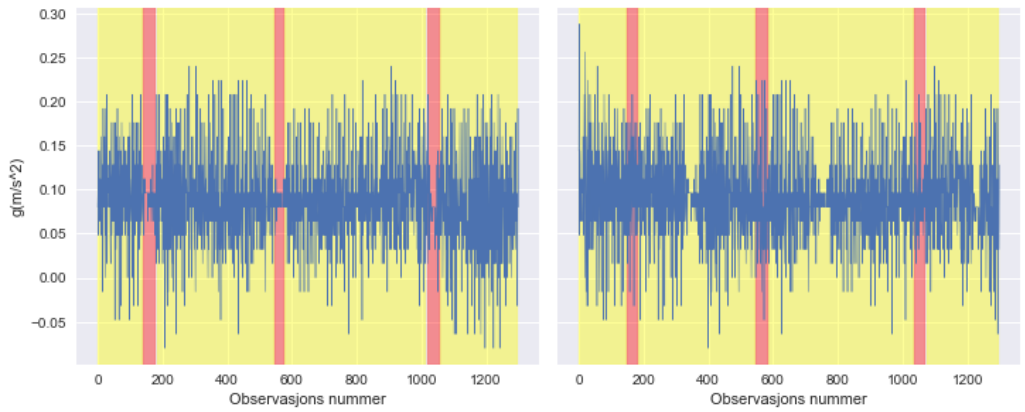
Figur 3.5: Akselerometer signal visualisert for fler-klasset bevegelse, med klave ID 37368 (Rosa egenkalv).

Figur 3.6 representerer datasettet som brukes til drøvtygge klassifiseringen. Fargen gul tilsvarer tygging og rød når kalven i stopper å tygge for å svelge/gulpe drøv.



Figur 3.6: Akselerometer signal visualisert for binær drøvtygge klassifisering, med klave ID 35396 (Hvit egenkalv).

I tillegg til korreksjon for tidssonene blir drøvtygge datasettet forskjøvet 19 sekunder, dette for å få god korrespondanse mellom atferdene tygge/ikke-tygge og akselerometer verdien. Figur 3.7 viser tydelig at atferds observasjonene er synkronisert med akselerometer signalet etter korreksjonen. Feilen ble oppdaget ved å analysere dataene før maskinlæringen, og rettet ved å manuelt justere antall sekund forskyving til signalene matcher atferden. Resultatet av dette belyser viktigheten ved å alltid analysere og sette seg inn i dataene man skal bruke før man starter med maskinlæring.



Figur 3.7: Sammenligning før og etter korreksjon. Kun x-akse og 1300 akselerometer observasjoner for binær drøvtygge klassifisering, med klave ID 35396 (Hvit egenkalv).

Etter sammenkoblingen sitter man igjen med to datagrunnlag, en for bevegelses klassifisering og en for klassifisering av drøvtygging. Antall observasjoner for datagrunnlaget bevegelse:

- 35396 (Hvit egenkalv): 53262
- 37368 (Rosa egenkalv): 49345

Antall observasjoner for datagrunnlaget drøvtygging:

- 35396 (Hvit egenkalv): 13346

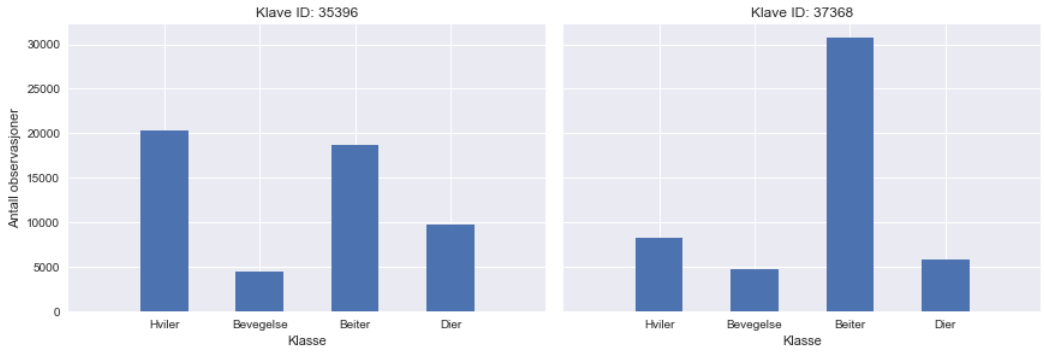
3.5.4 Dataoppdeling - trening, validering og test datasett

I en enkel maskinlærings-oppgave med styrt-læring er minimum kravet at datasettet deles inn i et trenings- og testsett. Dette er gjort for å kunne teste modellen senere på usette data som gir en indikasjon på hvor generalisert modellen er, så det er viktig at testsettet ikke er en del av treningssettet. I denne studien blir det gjort parameter-innstillinger på modellen og de ulike innstillingene må testes på egne usette data for å kunne sammenlignes, det blir derfor tildelt en egen del av datasettet til validering.

Akselerometer signal observasjonene er målt over tid og har derfor temporale avhengigheter med hverandre som gjør at de ikke kan splittes tilfeldig. Datasettene for bevegelse og drøvtygging blir splittet forskjellig. Fler- og binærklasse bevegelses datasettene inneholder observasjoner fra to kalver og datasettet for drøvtygging har data kun fra en kalv. For å kunne observere hvilke utslag valg av datasplitting har for modellene blir det gjort to ulike oppsplittinger. Valgene tatt ved oppsplittingene for trening, validering og test datasettene er basert på hva som gir gode klassefordelinger samtidig som at tidsstemplingen til signalene for hvert sett er kontinuerlige uten oppdelinger.

Flerklasset bevegelse

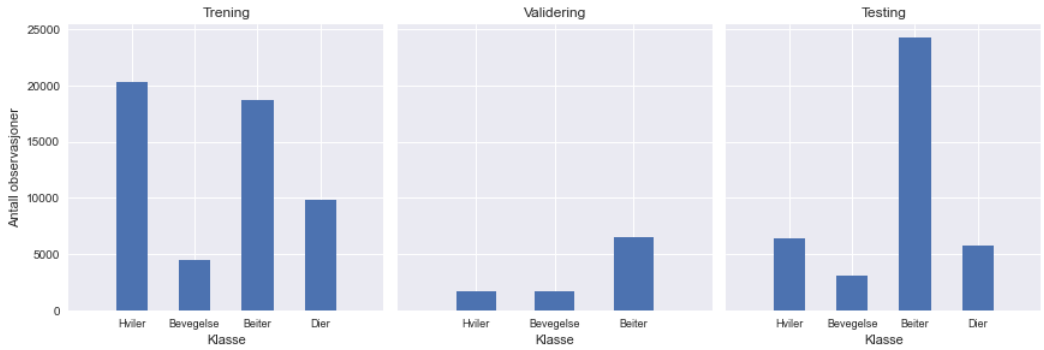
Figur 3.8 viser klassebalansen for bevegelsene til begge kalvene og man kan se at enkelte bevegelser er underrepresentert. Det optimale i en maskinlærings-modell er å ha mest mulig lik fordeling, men samtidig ha en distribusjon som på best mulig måte representerer den generelle oppførselen til kalven.



Figur 3.8: Klassebalanse for flerklasset datasett med Hvit egenkalv (35396) og Rosa egenkalv (37368).

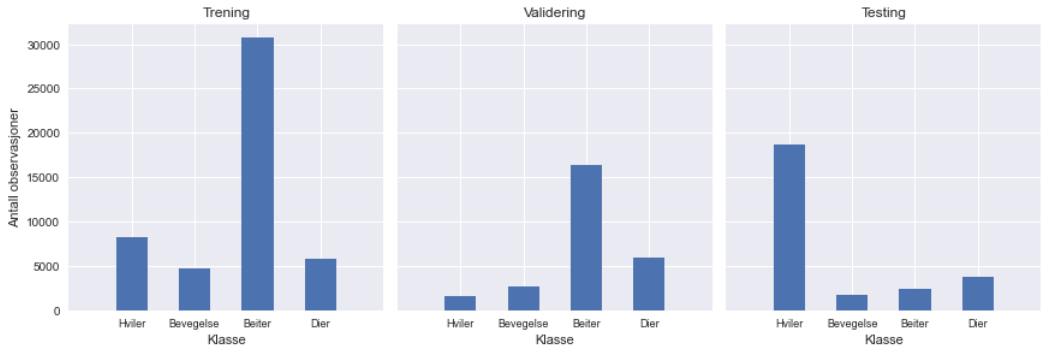
I første oppsplitting blir alle observasjoner fra Hvit egenkalv brukt til trenings-datasettet. De første 20% av Rosa egenkalv observasjonene blir satt av til validering, og resterende 80% til test-datasettet. Figur 3.9 viser den resulterende klassebalansen etter denne oppdelingen, og gir en ganske god fordeling for trenings-datasettet. Men man kan observere at diing ikke er med i valideringen, dette er på grunn av vanskeligheten ved å dele opp tidsserier ved små datagrunnlag.

Test distribusjonen viser et relativt høyt antall observasjoner for beiting, dette gir mest sannsynlig kunstig høy nøyaktighet ved predikering om man ikke bruker nøyaktighets-mål som F1-score.



Figur 3.9: Klassebalanse for flerklasset datasett, oppdeling 1.

I andre oppsplitting blir alle observasjoner fra Rosa egenkalv brukt til trenings datasettet. Den første halvdel av Hvit egenkalv blir brukt til validering, og andre halvdel til test-datasettet. Figur 3.10 viser klassebalansen etter oppdelingen, og viser relativt mange observasjoner for beiting. Valideringen er i dette tilfellet representert av alle fire bevegelser, som er bra. Test datasettet inneholder mange observasjoner for hvile, som kan gi utslag i form av dårlig nøyaktighet ved predikeringen på grunn av klasse ubalanse.

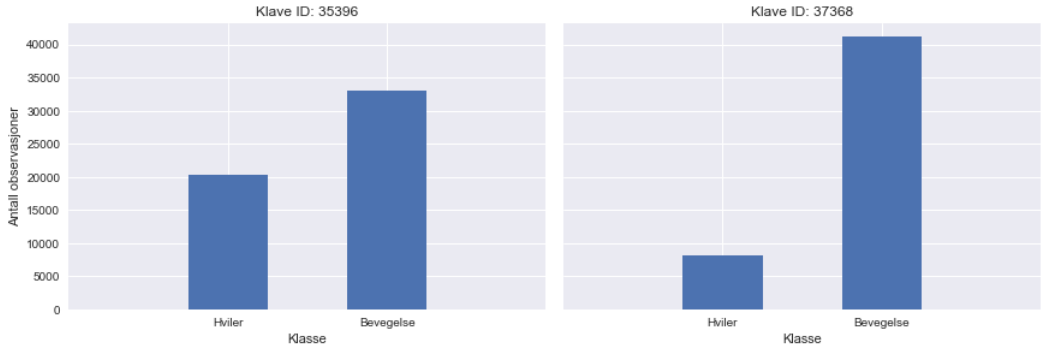


Figur 3.10: Klassebalanse for flerklasset datasett, oppdeling 2.

Binærklasset bevegelse

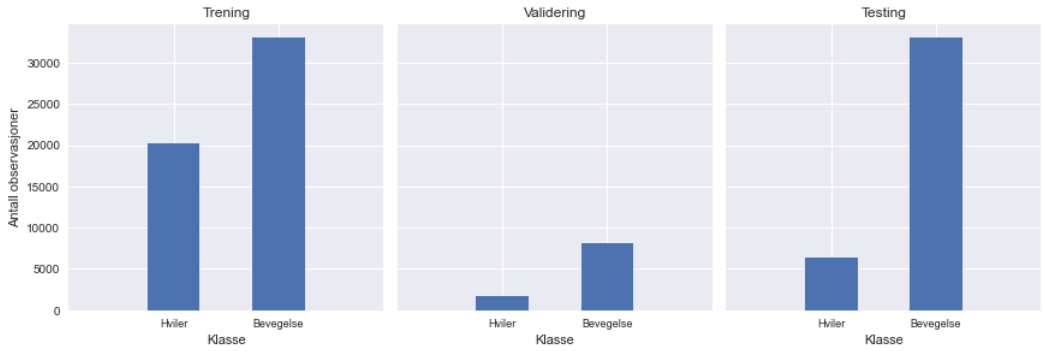
For å få til en binær klassifisering av bevegelsene må de fire ordinære klassene sammenslås til to, derfor blir klassene diing og beiting omgjort til klassen bevegelse.

Figur 3.11 viser balansen for klassene til begge kalvene, Hvit egenkalv har en grei fordeling og Rosa egenkalv har større representasjon av klassen bevegelse.

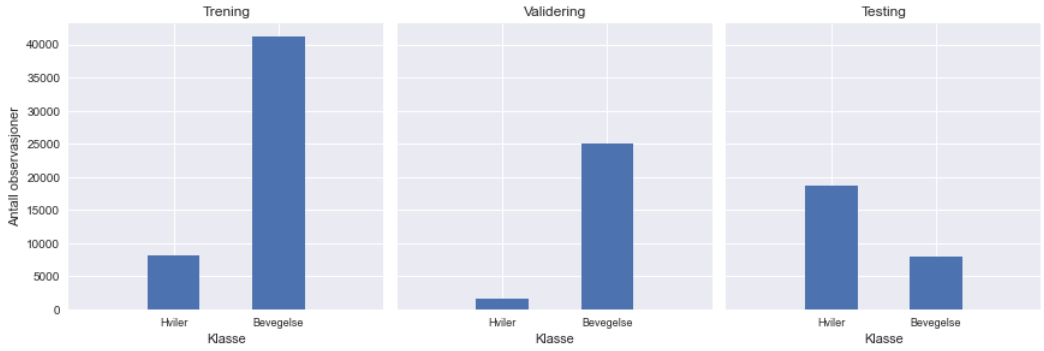


Figur 3.11: Klassebalanse for binærklasset datasett med Hvit egenkalv (35396) og Rosa egenkalv (37368).

Oppdeling 1 og 2 for den binære bevegelses klassifiseringen bruker samme oppdelings-strategi som hos den fler-klassede, som resulterer i Figurene 3.12 og 3.13. Alle settene inneholder observasjoner fra begge klasser. Oppdeling 1 har en mer balansert fordeling enn oppdeling 2 på trenings-dataene, men det motsatte kan observeres for test-dataene.



Figur 3.12: Klassebalanse for binærklasset datasett, oppdeling 1.

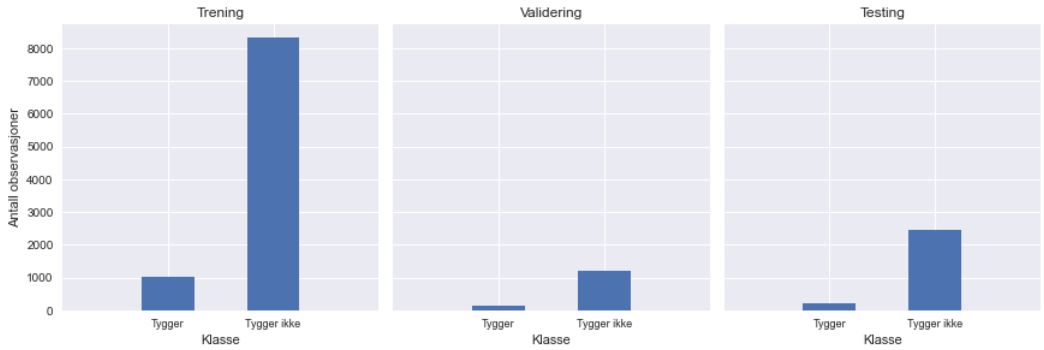


Figur 3.13: Klassebalanse for binærklasset datasett, oppdeling 2.

Drøvtygging

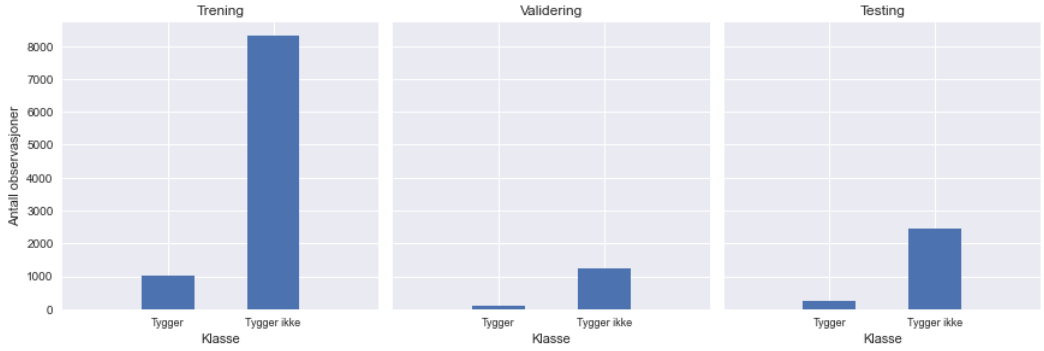
Datasettet for drøvtygging inneholder kun observasjoner fra Hvit egenkalv og er derfor delt i tre for å få trening, validering og test-datasett med totalt 13346 observasjoner. Siden kalven bruker noen få sekunder på å svelge/tygge drøv blir naturligvis denne klassen underrepresentert med kun 1360 observasjoner, og andelen tygging har mange flere med 11986 observasjoner.

I oppdeling 1 så er de første 70% tildelt trenings-settet, påfølgende 10% til validering og resterende 20% til test-datasettet, resultatet av oppdelingen er vist i Figur 3.14.



Figur 3.14: Klassebalanse for drøvtygge datasett, oppdeling 1.

I oppdeling 2 så er de første 20% tildelt test-settet, påfølgende 10% til validering og resterende 70% til trenings-datasettet, og er en omvending av oppdeling 1. Resultatet av oppdelingen er vist i Figur 3.15.



Figur 3.15: Klassebalanse for drøvtygge datasett, oppdeling 2.

3.5.5 Standardisering og danne sekvenser

Etter oppdelingen av dataene til trening-, validering- og test-datasett blir de standardisert. Standardiseringen baserer seg på å trekke fra gjennomsnittet og skalere mot enhet variansen, dette sentrerer dataene rundt null som er ønskelig i maskinlæring.

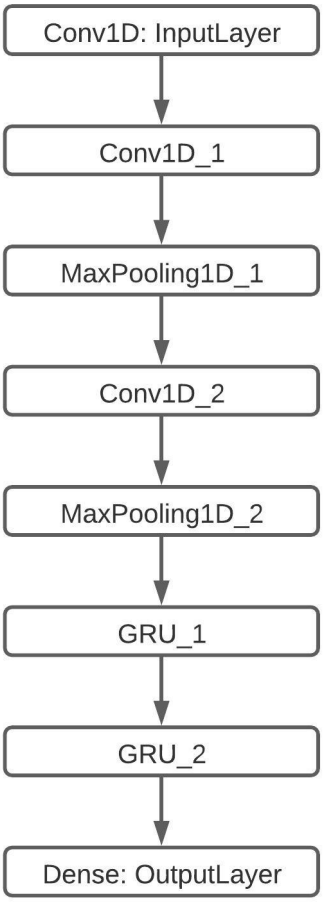
Før dataene er klar for å settes inn i nettverks-modellene må det dannes sekvenser. Antall observasjoner i sekvensen og antall steg det skal være mellom hver sekvens er to parametre som blir bestemt ut ifra hvilke parameter den best presterende modellen har.

3.6 Nettverksmodell

Det er i denne studien valgt å bruke to forskjellige nettverks-modeller basert på to kjente metoder innen tilbakevendende nevrale nettverk. Den ene strukturen bygger videre på Gated Recurrent Unit arkitekturen og den andre på Long Short Term Memory. Nettverks-modellene er like for alle datasett, men de vil ha ulike parametere bestemt av den beste modellen fra parameter-innstillingen.

3.6.1 Nettverksmodell basert på Gated Recurrent Unit (GRU)

Valget av å basere en av modellene på GRU er begrunnet med at den er kjent for å gi gode resultater på mindre datasett og er raskere enn andre tilbakevendende nettverks arkitekturer [24]. GRU er forholdsvis også en ganske moderne arkitektur innen tilbakevendende nevrale nettverk som er interessant å prøve ut sammen med sensor data.



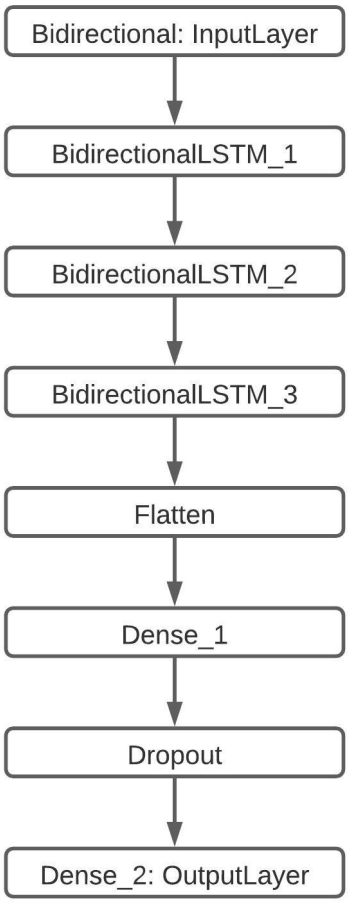
Figur 3.16: Modell strukturen hvor det er brukt GRU.

Figur 3.16 viser modellen som inneholder totalt åtte lag, hvor to av de er GRU. Første laget i modellen etter inngangslaget er et konvensjonelt-lag med en dimensjon. Dette laget har en kjernestørrelse på tre og bruker ReLu som aktiveringsfunksjon. Antall filter brukt i konvuleringen er en parameter som blir innstilt med ulike verdier. Dataene fra konvulsjonen går deretter inn i et samle-lag med en samle-størrelse på fire og *same padding*. Kombinasjonen av et konvulsjons-lag og samle-lag blir gjentatt, bare med en samle-størrelse på to i stedet for fire. De resterende to lagene før det siste predikerings-laget er GRU. Antall enheter i lagene bestemmes ved innstilling av parameterne, men alle GRU-lagene bruker tanh som aktiveringsfunksjon. Det siste predikerings-laget har like mange noder som antall klasser som skal predikeres, og med softmax aktiveringsfunksjon. Hele modellen bruker Adam som optimaliserer hvor lærings-raten er innstilt i parameter innstillingen.

3.6.2 Nettverksmodell basert på Long- Short Term Memory (LSTM)

Den andre nettverks-arkitekturen baserer seg på LSTM, som er en eldre og mer utprøvd modell. Modellen er kjent for å kunne holde på viktig tidligere informasjon over lengre perioder, men

kan slite med å lære ved mindre datasett. Nettverks-arkitekturen vil gi mulighet for å gjøre en interessant sammenlikning av prestasjon med andre arkitekturer.



Figur 3.17: Modell strukturen hvor det er brukt LSTM.

Figur 3.17 viser modellen som inneholder totalt åtte lag, hvor tre av de er LSTM. Disse tre lagene er plassert rett etter inngangslaget og har ulike enheter basert på parameter-innstilling. Etter disse LSTM lagene er det lagt inn et dimensjons reduserende lag som slår sammen sekvensene det siste LSTM laget gir ut. Disse sekvensene blir gitt videre til et vanlig nevralt lag med aktiveringsfunksjonen ReLu og antall noder basert på ulike verdier fra parameter-innstilling. Et lag med utkasting er lagt inn før predikeringen for å regularisere modellen, utkastings-verdien blir også her tunet med ulike verdier. Siste predikerings-laget er her lik den i GRU modellen.

3.6.3 Parameter innstillinger for modell

Det finnes uendelige forskjellige kombinasjoner av type og antall lag i et nevralt nettverk samt dens parameterverdier. Det er derfor i dette studiet valgt å forholde seg til den gitte strukturen på modellene, men samtidig prøve ulike kombinasjoner av parameter verdier. For hver kombinasjon av parameterverdier blir det konstruert og lært opp en modell, dette vil si at jo flere parameterverdier som blir valgt jo lengre tid tar det å teste alle modellene. For å ikke lekke informasjon over til test settet blir hver modell kombinasjon trent opp på trenings-settet og testet på validerings-settet.

Parametre - Fler- og binærklasse bevegelse

Parametere brukt sammen med GRU modell:

- Antall observasjoner per sekvens: [32, 64, 96]
- Observasjons steg mellom sekvenser: [15, 31, 62]
- Antall filter i konvulsjons lag: [10, 20, 50, 100]
- Antall enheter i GRU: [16, 32, 64, 96]
- Lærings rate: [0.001, 0.0001]
- Batch størrelse: [32, 64, 128]

Dette resulterer i 864 forskjellige modeller som trenes og testes.

Parametere brukt sammen med LSTM modell:

- Antall observasjoner per sekvens: [32, 64, 96]
- Observasjons steg mellom sekvenser: [15, 31, 62]
- Antall enheter i LSTM: [64, 96, 128]
- Prosent utkasting: [0, 0.2]
- Antall nevroner i lag: [4, 6, 10]
- Lærings rate: [0.001, 0.0001]
- Batch størrelse: [32, 64, 128]

Dette resulterer i 648 forskjellige modeller som trenes og testes.

Parametre - Drøvtygging

Parametere brukt sammen med GRU modell:

- Antall observasjoner per sekvens: [11, 32, 64]
- Observasjons steg mellom sekvenser: [5, 10, 16, 31]
- Antall filter i konvulsjons lag: [10, 50, 100]
- Antall enheter i GRU: [16, 32, 64, 96]
- Lærings rate: [0.001, 0.0001]
- Batch størrelse: [32, 64, 128]

Dette resulterer i 864 forskjellige modeller som trenes og testes.

Parametere brukt sammen med LSTM modell:

- Antall observasjoner per sekvens: [11, 32, 64]
- Observasjons steg mellom sekvenser: [5, 10, 16, 31]
- Antall enheter i LSTM: [16, 32, 64, 96]

- Prosent utkasting: [0, 0.2]
- Antall nevroner i lag: [3, 6, 10]
- Lærings rate: [0.001, 0.0001]
- Batch størrelse: [32, 64, 128]

Dette resulterer i 1728 forskjellige modeller som trenes og testes.

Vurdering av modellene

Ved trening av alle disse modellene blir alltid den modellen med lavest taps-verdi på valideringssettet lagret, dette for å unngå modeller som kun har memorert trenings settet og ikke generaliserer godt nok. De to beste modellene blir lagret. Siden det er tre datasett med to forskjellig oppdelinger som blir trent opp med både en GRU og LSTM modell ender man opp med totalt 24 lagrede modeller. Disse modellene kan brukes på test-datasetter for å hente ut resultater.

Antall epoker modellene skal kjøre er fastsatt som en parameter. For å automatisk stoppe treningen av modellene når taps-verdien ikke lengre forbedres er det brukt en innebygget funksjon kalt *EarlyStopping*. Fordelen ved denne funksjonen er at man slipper å kjøre modellen med ulike verdier for antall epoker. Selv om man da setter en høy verdi for antall epoker så vil modellen stoppe å trene når den ikke lengre gir forbedrede resultater, som gjør at man unngår overtrening av dataene.

4 Resultater

I dette kapittelet blir forvirringsmatriser fra resultatene vist, de to beste modeller fra hver kombinasjon av datasett, oppdeling og type modell, til sammen 24 forskjellige modeller. For å få et best mulig mål på den helhetlige nøyaktigheten av modellene er det også vist tabeller som inneholder recall og F1-score i Tillegg A

4.1 Flerklasse bevegelse predikering

4.1.1 GRU

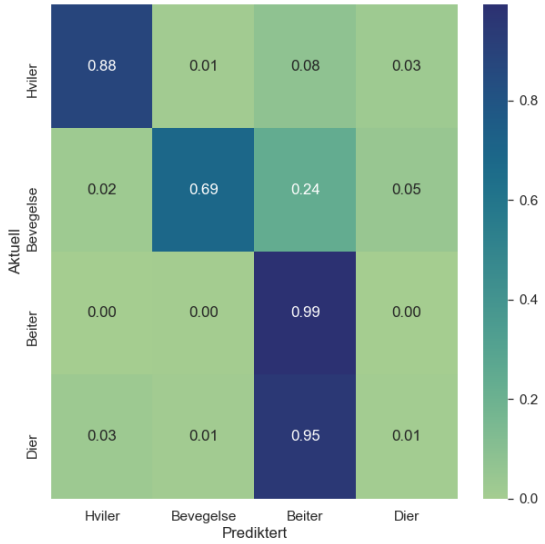
Tabell 4.1: Best modell parametre til resultat for flerklasset GRU.

Parameter	Best oppd. 1	Nest best oppd. 1	Best oppd. 2	Nest best oppd. 2
Sekvens-størrelse	96	96	96	96
Sekvens-steg	15	15	31	62
Antall filter	20	50	100	50
Antall enheter	64	64	16	16
Lærings rate	0.001	0.001	0.001	0.001
Batch størrelse	32	32	64	32
F1-score	0.747	0.726	0.658	0.640

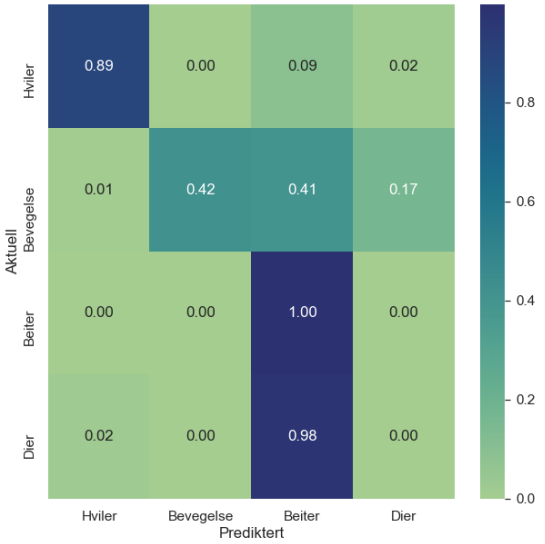
Ut ifra samlet F1-score vist i Tabell 4.1 er det oppdeling 1 som gir best resultat. Om man ser på forvirringsmatrisene for oppdeling 1 og 2 vist i Figurene 4.1 til 4.4 så kan man observere bedre resultat for nesten alle atferdene hos oppdeling 1.

Oppdeling 1

I Figurene 4.1 og 4.2 som er resultatet for de to beste modellene kan man observere at begge sliter med å klassifisere atferden dier. Den beste modellen representert av Figur 4.1 klarer å klassifisere bevegelse bedre enn den nest beste i Figur 4.2, men gjør det litt dårligere på atferden hviler og beiter. Ut i fra parameterforskjellene vist i Tabell A.2.1 så skyldes forskjellen antall filter i konvolusjons-laget. En forklaring kan være at siden den beste modellen har mindre antall filter, som gir en bedre generalisering for modellen.



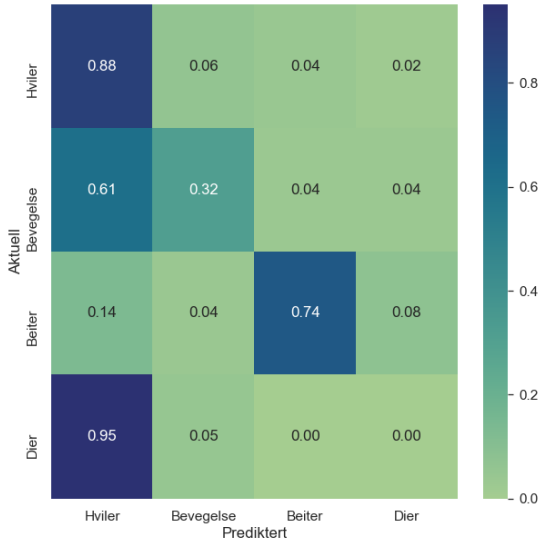
Figur 4.1: Forvirringsmatrise flerklasset bevegelse, GRU med oppdeling 1. Resultat fra beste modell på testdata.



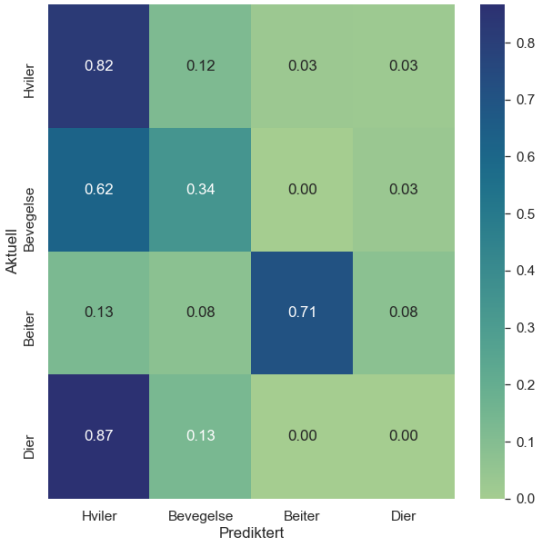
Figur 4.2: Forvirringsmatrise flerklasset bevegelse, GRU med oppdeling 1. Resultat fra nest beste modell på testdata.

Oppdeling 2

I Figurene 4.3 og 4.4 kan man observere at ingen av modellene klarer å klassifisere atferden dier korrekt. Den nest beste modellen vist av Figur 4.4 klarer å klassifisere bevegelse en liten grad bedre enn den beste, men gjør det dårligere i atferdene hviler og beiter.



Figur 4.3: Forvirringsmatrise flerklasset bevegelse, GRU med oppdeling 2. Resultat fra beste modell på testdata.



Figur 4.4: Forvirringsmatrise flerklasset bevegelse, GRU med oppdeling 2. Resultat fra nest beste modell på testdata.

4.1.2 LSTM

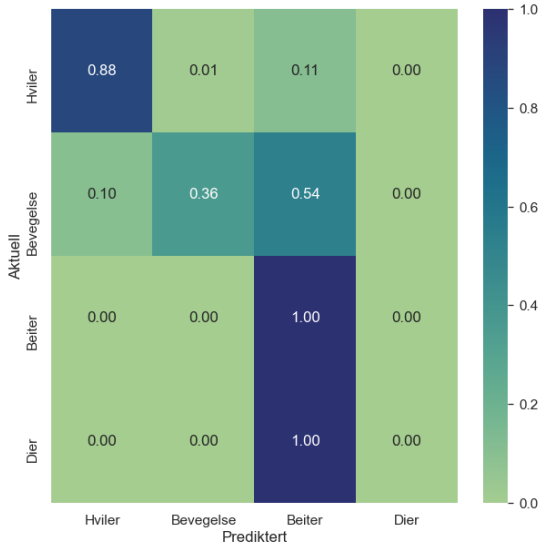
Tabell 4.2: Best modell parametre til resultat for flerklasset LSTM.

Parameter	Best oppd. 1	Nest best oppd. 1	Best oppd. 2	Nest best oppd. 2
Sekvens-størrelse	64	64	32	96
Sekvens-steg	15	15	62	62
Antall enheter i LSTM	64	96	64	96
Prosent utkasting	0	0.2	0.2	0.2
Antall nevroner	10	10	6	4
Lærings rate	0.001	0.001	0.0001	0.001
Batch størrelse	64	64	128	32
F1-score	0.714	0.702	0.655	0.645

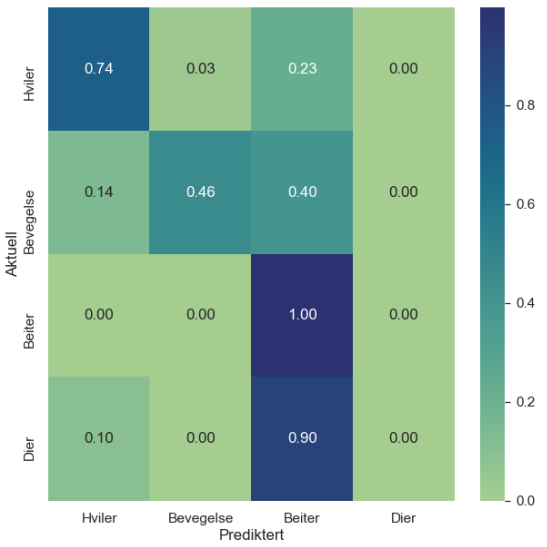
Ut ifra samlet F1-score vist i Tabell 4.2 er det oppdeling 1 som gir best resultat. Om man ser på forvirringsmatrisene for oppdeling 1 og 2 vist i Figurene 4.5 til 4.8 så kan man observere bedre resultat for atferden beiter hos oppdeling 1. Oppdeling 2 klarer ikke å klassifisere atferden bevegelse og ingen av oppdelingene klarer å klassifisere atferden dier korrekt.

Oppdeling 1

I Figurene 4.5 og 4.6 som er resultatet for de to beste modellene kan man observere at begge sliter med å klassifisere atferden bevegelse, og ikke klarer å klassifisere dier. Den beste modellen representert av Figur 4.5 og den nest beste Figur 4.6 klarer begge å klassifisere atferden beiter korrekt. Den neste beste modellen gjør det bedre på atferden bevegelse, men dårligere på hviler.



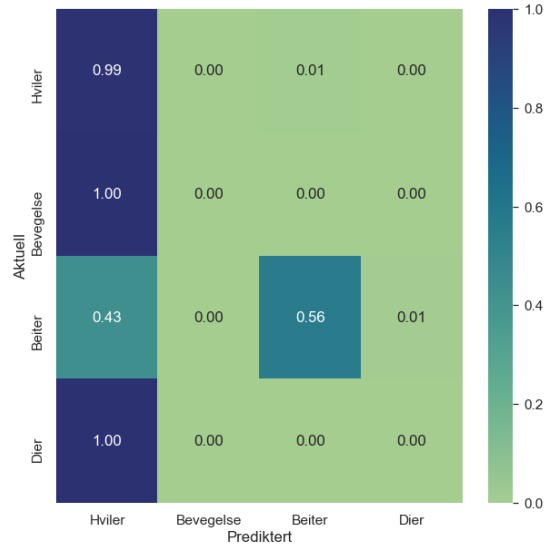
Figur 4.5: Forvirringsmatrise flerklasset bevegelse, LSTM med oppdeling 1. Resultat fra beste modell på testdata.



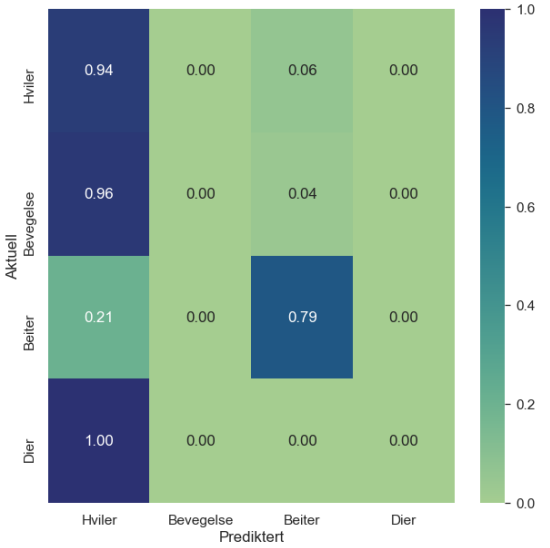
Figur 4.6: Forvirringsmatrise flerklasset bevegelse, LSTM med oppdeling 1. Resultat fra nest beste modell på testdata.

Oppdeling 2

I Figurene 4.7 og 4.8 kan man observere at ingen av modellene klarer å klassifisere verken bevegelse eller dier korrekt. Den nest beste modellen vist av Figur 4.8 klarer å klassifisere beiter bedre enn den beste, men gjør det litt dårligere i atferdene hviler. Som kan skyldes de ulike størrelsene på sekvensene.



Figur 4.7: Forvirringsmatrise flerklasset bevegelse, LSTM med oppdeling 2. Resultat fra beste modell på testdata.



Figur 4.8: Forvirringsmatrise flerklasset bevegelse, LSTM med oppdeling 2. Resultat fra nest beste modell på testdata.

4.1.3 Kort sammendrag - Flerklasset bevegelse

For flerklasset bevegelse er det modellen GRU med oppdeling 1 som gir høyest totale F1-score av alle modellene med en score på 0.747, forvirringsmatrisen for denne modellen er vist i Figur 4.1. Ingen av modellene klarer å klassifisere diing korrekt, bortsett fra den beste modellen som klarer å klassifisere 1% riktig.

Modell med beste F1-score fordelt på atferd:

- Hvile: Nest beste modell for GRU - oppdeling 1
- Bevegelse: Beste modell for GRU - oppdeling 1
- Beiter: Beste modell for GRU - oppdeling 1
- Dier: Beste modell for GRU - oppdeling 1

4.2 Binær bevegelse predikering

4.2.1 GRU

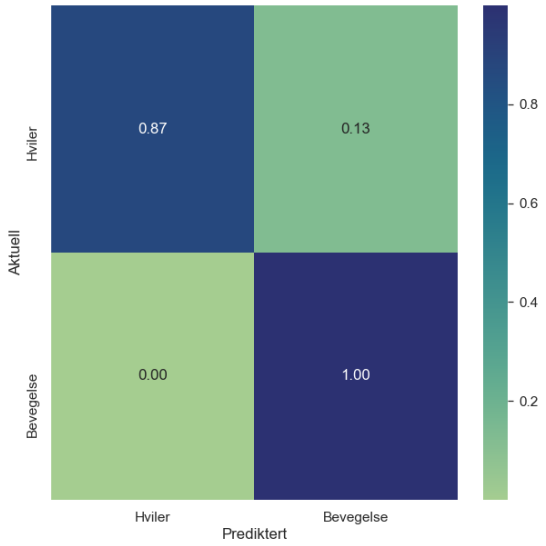
Tabell 4.3: Best modell parametre til resultat for binærklasset GRU.

Parameter	Best oppd. 1	Nest best oppd. 1	Best oppd. 2	Nest best oppd. 2
Sekvens-størrelse	96	96	96	96
Sekvens-steg	15	15	62	62
Antall filter	20	100	100	100
Antall enheter	32	96	16	16
Lærings rate	0.001	0.001	0.001	0.0001
Batch størrelse	32	32	32	32
F1-score	0.979	0.976	0.703	0.696

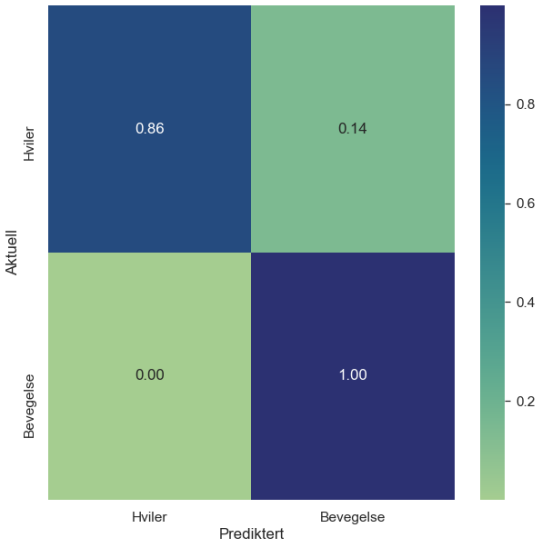
Ut ifra samlet F1-score vist i Tabell 4.3 er det oppdeling 1 som gir best resultat. Om man ser på forvirringsmatrisene for oppdeling 1 og 2 vist i Figurene 4.9 til 4.12 så kan man observere bedre resultat for atferden bevegelse ved oppdeling 1. Oppdeling 1 gjør en nesten feilfri klassifisering av begge atferdene. Oppdeling 2 gjør en bra klassifisering av atferden hviler, men gjør det dårlig på atferden bevegelse.

Oppdeling 1

Både den beste og den nest beste modellen klarer å predikere atferden bevegelse feilfritt.



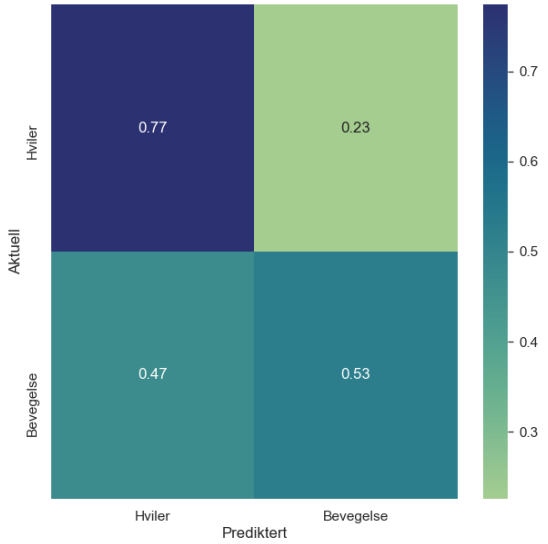
Figur 4.9: Forvirringsmatrise binærklasset bevegelse, GRU med oppdeling 1. Resultat fra beste modell på testdata.



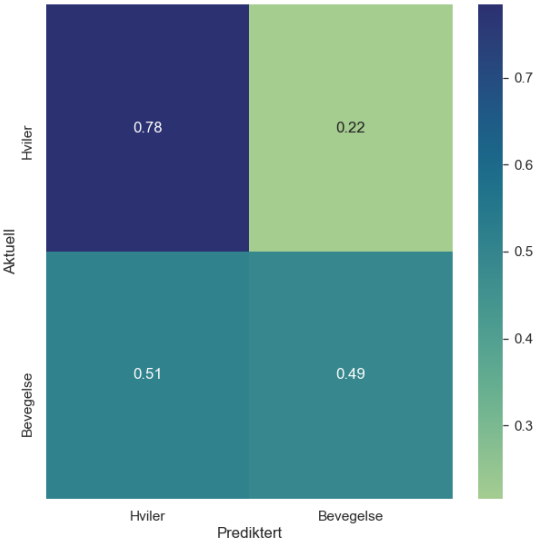
Figur 4.10: Forvirringsmatrise binærklasset bevegelse, GRU med oppdeling 1. Resultat fra nest beste modell på testdata.

Oppdeling 2

Den beste modellen har en litt bedre score for atferden bevegelse, men en litt dårligere score for atferden hviler.



Figur 4.11: Forvirringsmatrise binærklasset bevegelse, GRU med oppdeling 2. Resultat fra beste modell på testdata.



Figur 4.12: Forvirringsmatrise binærklasset bevegelse, GRU med oppdeling 2. Resultat fra nest beste modell på testdata.

4.2.2 LSTM

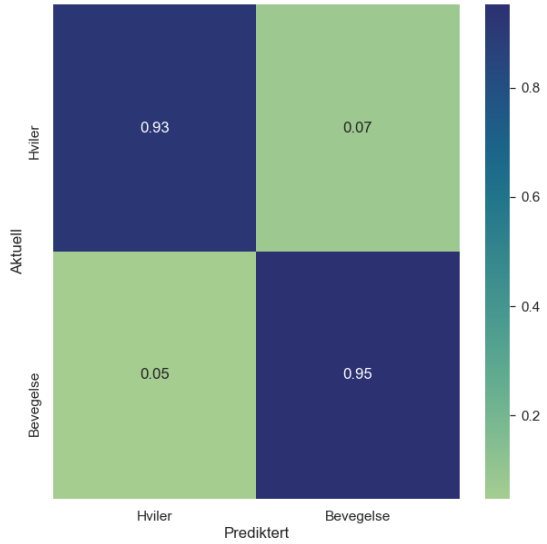
Tabell 4.4: Best modell parametre til resultat for binærklasset LSTM.

Parameter	Best oppd. 1	Nest best oppd. 1	Best oppd. 2	Nest best oppd. 2
Sekvens-størrelse	96	96	96	96
Sekvens-steg	31	31	31	62
Antall enheter i LSTM	96	64	64	96
Prosent utkasting	0	0.2	0.2	0.2
Antall nevroner	10	4	4	4
Lærings rate	0.001	0.001	0.0001	0.001
Batch størrelse	192	128	192	32
F1-score	0.950	0.938	0.690	0.688

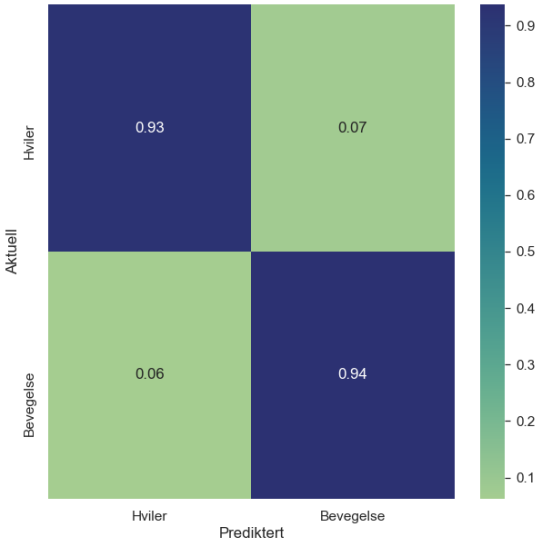
Ut ifra samlet F1-score vist i Tabell 4.4 er det oppdeling 1 som gir best resultat. Om man ser på forvirringsmatrisene for oppdeling 1 og 2 vist i Figurene 4.13 til 4.16 så kan man observere at oppdeling 1 har et bedre resultat for begge atferdene. Oppdeling 1 gjør en nesten feilfri klassifisering av atferden bevegelse.

Oppdeling 1

Den beste og nest beste modellen er ganske like, men den beste modellen gjør en litt bedre jobb med å klassifisere atferden bevegelse.



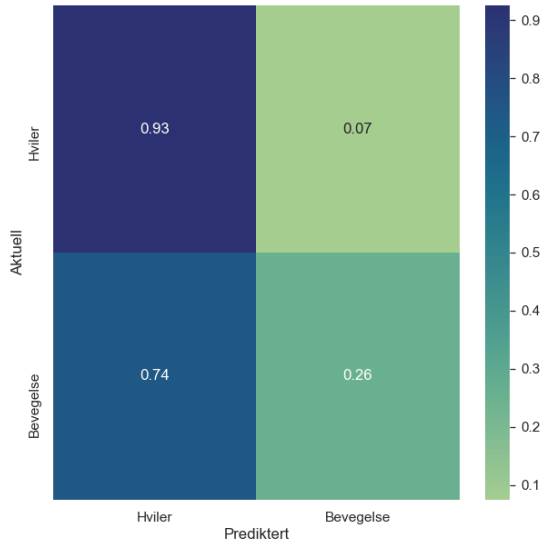
Figur 4.13: Forvirringsmatrise binærklasset bevegelse, LSTM med oppdeling 1. Resultat fra beste modell på testdata.



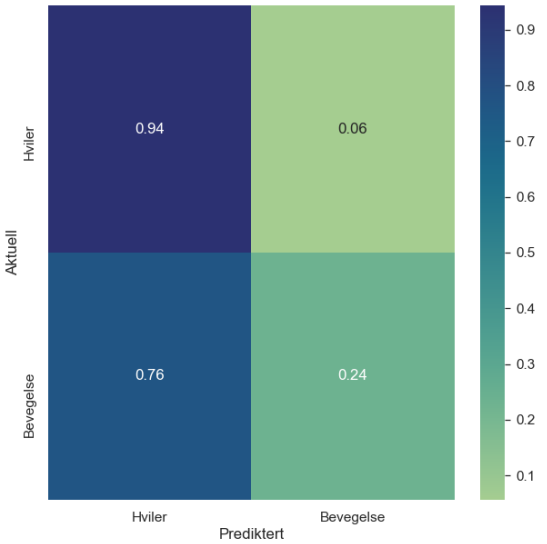
Figur 4.14: Forvirringsmatrise binærklasset bevegelse, LSTM med oppdeling 1. Resultat fra nest beste modell på testdata.

Oppdeling 2

Den beste modellen gjør en litt bedre klassifisering for atferden bevegelse, men litt dårligere for hviler.



Figur 4.15: Forvirringsmatrise binærklasset bevegelse, LSTM med oppdeling 2. Resultat fra beste modell på testdata.



Figur 4.16: Forvirringsmatrise binærklasset bevegelse, LSTM med oppdeling 2. Resultat fra nest beste modell på testdata.

4.2.3 Kort sammendrag - Binær bevegelse

For binær bevegelse er det modellen GRU med oppdeling 1 som gir høyest totale F1-score av alle modellene med en score på 0.979, forvirringsmatrisen for denne modellen er vist i Figur 4.9. Denne modellen gjør det også best på begge atferdene sett på hver for seg.

4.3 Drøvtygge predikering

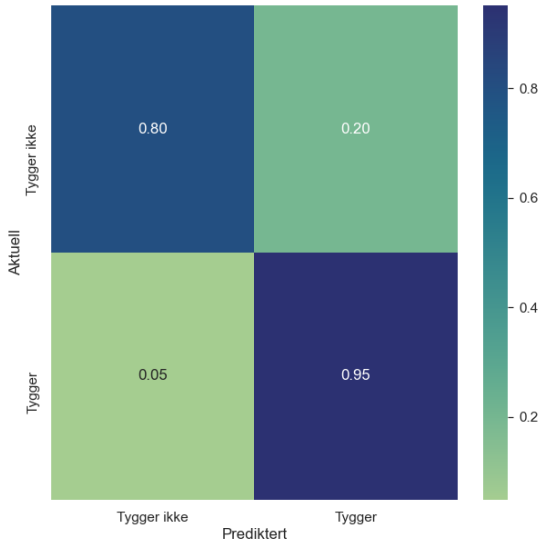
4.3.1 GRU

Tabell 4.5: Best modell parametre til resultat for drøvtygging GRU.

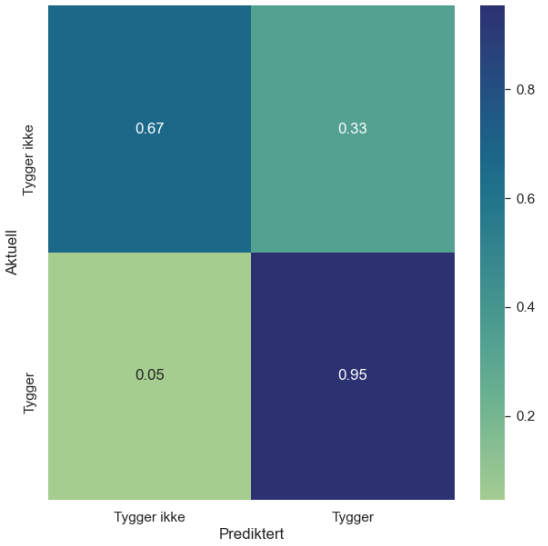
Parameter	Best oppd. 1	Nest best oppd. 1	Best oppd. 2	Nest best oppd. 2
Sekvens-størrelse	32	64	64	64
Sekvens-steg	31	16	5	10
Antall filter	10	100	100	50
Antall enheter	16	96	16	32
Lærings rate	0.001	0.001	0.001	0.001
Batch størrelse	64	32	64	32
F1-score	0.948	0.936	0.956	0.951

Ut ifra samlet F1-score vist i Tabell 4.5 er det oppdeling 2 som gir best resultat. Om man ser på forvirringsmatrisene for oppdeling 1 og 2 vist i Figurene 4.17 til 4.20 så kan man observere at oppdeling 2 har et bedre resultat for atferden tygger, men dårligere for ikke tygger. Oppdeling 2 får en bedre score fordi mesteparten av observasjonene har klassen tygger. Om man fokuserer på den underrepresenterte atferden tygger ikke, så er det oppdeling 1 som gjør det best.

Oppdeling 1

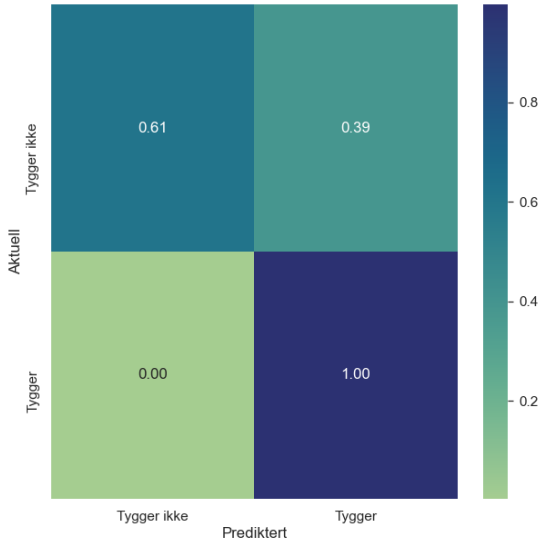


Figur 4.17: Forvirringsmatrise drøvtygging, GRU med oppdeling 1. Resultat fra beste modell på testdata.

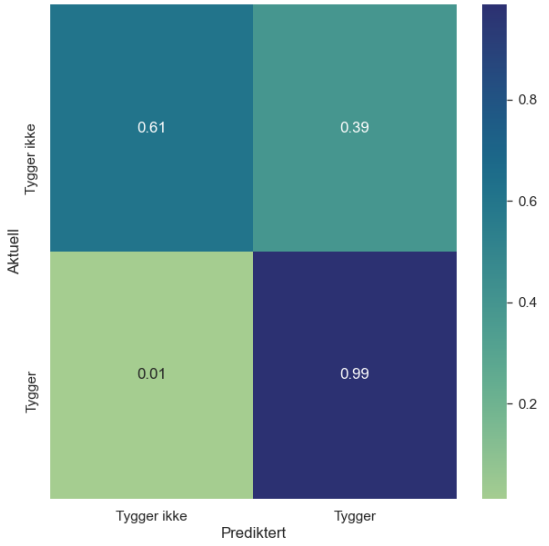


Figur 4.18: Forvirringsmatrise drøvtygging, GRU med oppdeling 1. Resultat fra nest beste modell på testdata.

Oppdeling 2



Figur 4.19: Forvirringsmatrise drøvtygging, GRU med oppdeling 2. Resultat fra beste modell på testdata.



Figur 4.20: Forvirringsmatrise drøvtygging, GRU med oppdeling 2. Resultat fra nest beste modell på testdata.

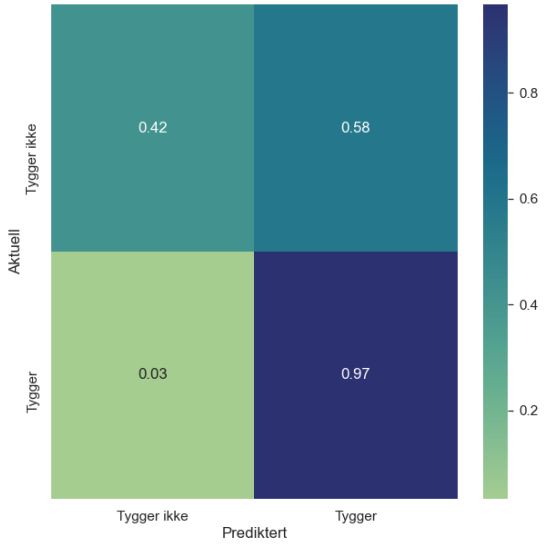
4.3.2 LSTM

Tabell 4.6: Best modell parametre til resultat for drøvtygging LSTM.

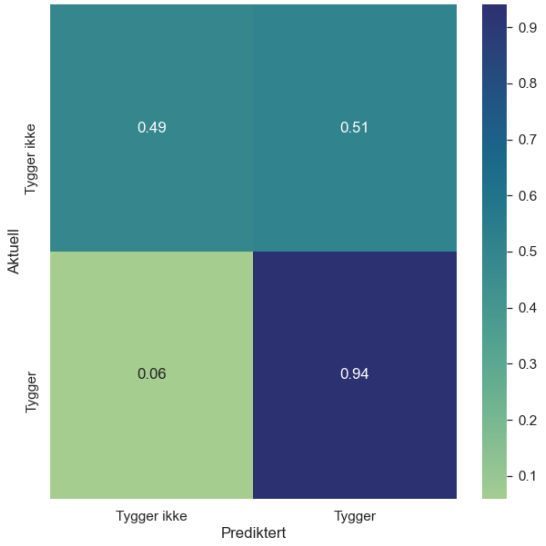
Parameter	Best oppd. 1	Nest best oppd. 1	Best oppd. 2	Nest best oppd. 2
Sekvens-størrelse	32	32	64	64
Sekvens-steg	5	5	31	31
Antall enheter i LSTM	16	64	64	64
Prosent utkasting	0	0	0.1	0
Antall nevroner	6	3	6	10
Lærings rate	0.001	0.001	0.001	0.0001
Batch størrelse	32	32	128	128
F1-score	0.918	0.907	0.861	0.861

Ut ifra samlet F1-score vist i Tabell 4.6 er det oppdeling 1 som gir best resultat. Om man ser på forvirringsmatrisene for oppdeling 1 og 2 vist i Figurene 4.21 til 4.24 så kan man observere at oppdeling 2 ikke klarer å klassifisere atferden tygger ikke. Oppdeling 1 klarer å klassifisere rundt halvparten av den underrepresenterte atferden tygger ikke.

Oppdeling 1

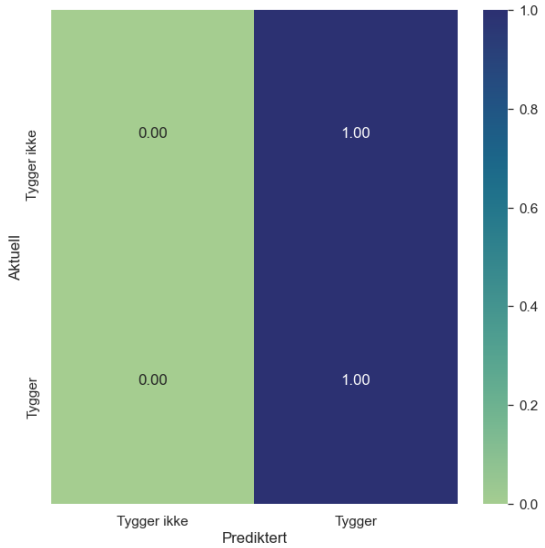


Figur 4.21: Forvirringsmatrise drøvtygging, LSTM med oppdeling 1. Resultat fra beste modell på testdata.

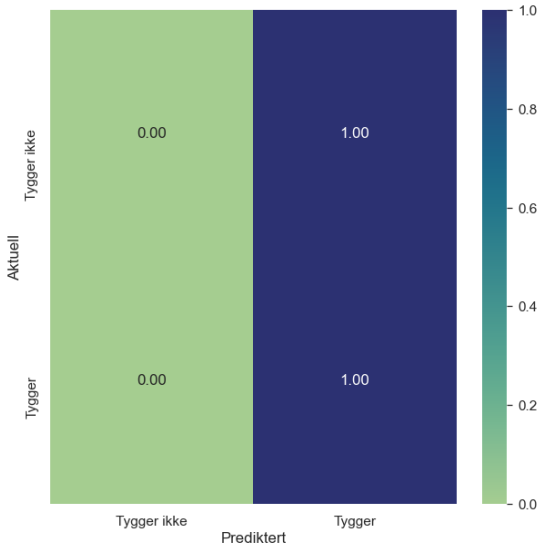


Figur 4.22: Forvirringsmatrise drøvtygging, LSTM med oppdeling 1. Resultat fra nest beste modell på testdata.

Oppdeling 2



Figur 4.23: Forvirringsmatrise drøvtygging, LSTM med oppdeling 2. Resultat fra beste modell på testdata.



Figur 4.24: Forvirringsmatrise drøvtygging, LSTM med oppdeling 2. Resultat fra nest beste modell på testdata.

4.3.3 Kort sammendrag - Drøvtygging

For drøvtygging er det modellen GRU med oppdeling 2 som gir høyest totale F1-score av alle modellene med en score på 0.956, forvirringsmatrisen for denne modellen er vist i Figur 4.19. Det er verdt å nevne at oppdeling 1 klarer å klassifisere den underrepresenterte klassen tygger ikke bedre enn oppdeling 2.

4.4 Kort sammendrag - Resultater

Den tilbakevendende nevrale nettverks arkitekturen som generelt sett presterer best over alle datasett og modeller er GRU, og den oppdelingen som gir best F1-score er oppdeling 1.

5 Diskusjon

Et kritisk blikk på arbeidet som er gjort hvor resultater, datasett og framgangsmåte diskuteres. Samt forslag til hva som kunne vært gjort bedre/annerledes.

5.1 Tolkning av resultatene

Forvirringsmatrisene gir ikke kun oversikt over den generelle prestasjonen til modellene, men viser også alle aspekter ved feilklassifiseringene. Resultatene visualisert av alle forvirringsmatrisene kombinert med informasjon om datasettet, klassebalansen og modellparametere gir ikke kun innsikt i tilbakevendende nevrale nettverks evne til atferds-klassifisering, men også verdifull informasjon som kan brukes for å forbedre modellene.

5.1.1 Tolkning - Flerklasset bevegelse

I forvirringsmatrisene for flerklasset bevegelse vist av Figurene 4.1 til 4.8 kan man se at modellene har en tendens til å klassifisere dier atferden som enten hviler eller beiter. Det er mest sannsynlig flere årsaker til denne feilklassifiseringen. Underrepresentasjon av klassen dier i forhold til beiting og hviler er en av grunnene til at den blir klassifisert som hviler eller beiter i modellene. Ubalansen kan observeres i klassebalanse histogrammene (Figurene 3.8 til 3.10) og i resultat tabellene vist i Tilleggene A.1.1 til A.1.8.

Grunnen for dårligere resultat av atferden bevegelse kan være den temporale lengden for hver bevegelse atferd i forhold til valgt sekvenslengde. Om man ser tilbake på Figurene 3.4 og 3.5 som viser hvor i datasettet de ulike klassene forekommer, så kan man se at observasjons-intervallene for hvert tilfelle av atferden bevegelse (lys blå) ikke er så store. Dette gjør at modellen ikke får nok etterfølgende sekvenser med atferden bevegelse til å lære, dette kan forverres ved lengre sekvenser. Siden læringen og predikeringen av sekvensene baserer seg på mange-til-en, så vil modellene predikere en atferd for hver gruppering (sekvens). Et godt eksempel som forklarer dette på en god måte er om man prøver å sette lengden på sekvensene lik lengden på testdatasettet. Da vil modellen klassifisere hele testdatasettet som en atferd, og alle underrepresenterte atferder vil bli feilklassifisert (usynlig for modellen).

Atferden dier blir ofte klassifisert som beiter, dette kan være på grunn av at akselerometer-signalene vist i Figurene 3.4 og 3.5 er for like til at modellen klarer å lære forskjellene. Og siden atferden beiting forekommer oftere enn dier så tror modellene at dier er beiter, og ikke motsatt.

Valget av oppdeling for datasettene ser ut til å påvirke hvilken klasse atferdene blir feilklassifisert. Dette er et tegn på for lite data og at modellene ikke klarer å generalisere godt med nye data. Data fra flere kalver hadde gitt modellene mulighet til å generalisere bedre under opplæring.

5.1.2 Tolkning - Binær bevegelse

I forvirringsmatrisene for binær bevegelse vist av Figurene 4.9 til 4.16 kan man se at modellene generelt sett klarer å predikere atferden hviler bedre enn bevegelse. Det er viktig å huske at for dette datasettet så er tre av atferdene fra flerklasset bevegelse slått sammen til atferden bevegelse, og atferden hviler er uendret. Så mange av argumentene fra flerklasset bevegelse gjelder også for dette datasettet.

Oppdeling 2 presterer dårligere både i GRU og LSTM. Klassebalansen er vist i Figurene 3.11 til 3.13 og man kan tydelig se at i oppdeling 2 er atferden bevegelse overrepresentert i trenings settet, og valideringen underrepresentert. Under modell treningen så er den beste modellen valgt ut ifra validerings-tapet, altså hvor god predikering modellen gjorde på validerings-settet. Dette kan være årsaken til at oppdeling 1 gjør en bedre klassifisering enn oppdeling 2.

5.1.3 Tolkning - Drøvtygging

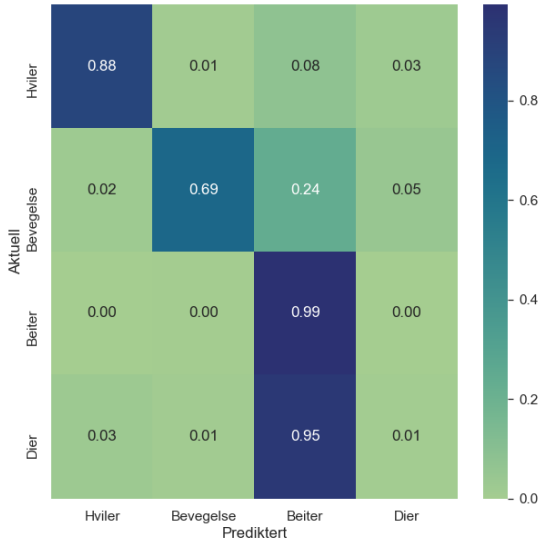
I forvirringsmatrisene for drøvtygging vist av Figurene 4.17 til 4.24 kan man se at alle modellene bortsett fra LSTM med oppdeling 2 klarer å klassifisere både Tygger og Tygger ikke. LSTM med oppdeling 2 klassifiserer alle observasjonene som Tygger. Klassebalansen er vist i Figurene 3.14 og 3.13, her kan man tydelig se at atferden Tygger ikke er underrepresentert for begge oppdelingene. I tillegg til at atferden Tygger ikke er underrepresentert så finner man også her problemet diskutert i flerklasset bevegelse med små observasjons-intervaller opp, som man kan observere i Figur 3.6.

5.2 Sammenlikning med ikke-styrt læring

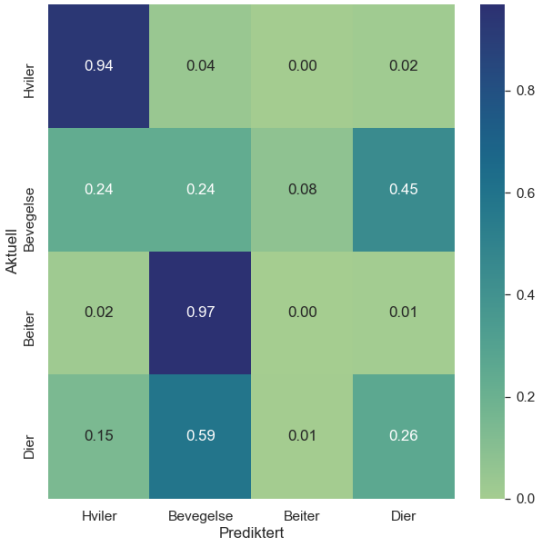
Ved å sammenlikne resultatene oppnådd i studien med en ikke-styrt læring modell så kan man trekke ut styrkene og svakhetene ved metodikken. Hvilke atferder klarer tilbakevendende nevrale nettverk å trekke ut fra dataene kontra *k-means clustering*, eller omvendt.

5.2.1 Sammenlikning - Flerklasset bevegelse

Flerklasset bevegelse datasettet med *k-means clustering* ga en F1-score på kun 0.175 vist i Figur 5.2, kontra beste modellen oppnådd i oppgaven på 0.747 vist i Figur 5.1. Den ikke-styrte lærings-metoden klarer å klassifisere noe av klassen dier korrekt, men bommer på klassen beiter. Dette er også grunnen til at den får en lav F1-score da klassen beiter er en viktig og betydelig del av datasettet. *K-means clustering* klarer her også å gjøre det bedre på klassen hviler.



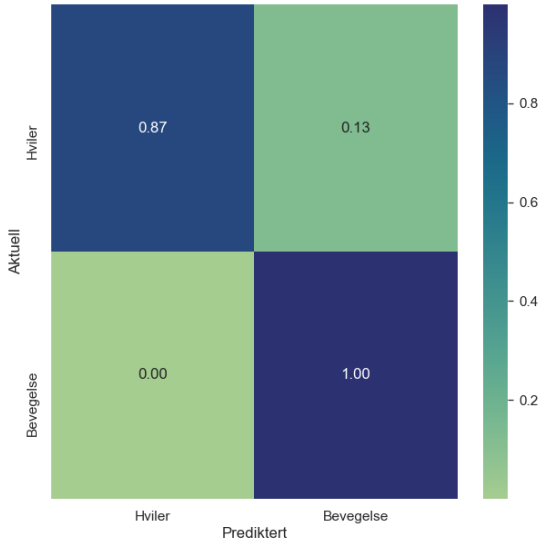
Figur 5.1: Beste resultat for flerklasset bevegelse, GRU med oppdeling 1.



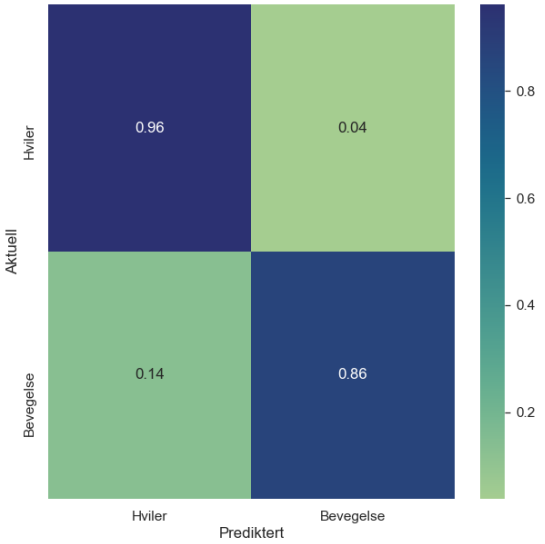
Figur 5.2: Forvirringsmatrise k-means clustering med 4 grupperinger.

5.2.2 Sammenlikning - Binær bevegelse

Binær bevegelse datasettet med *k-means clustering* ga en F1-score på 0.891 vist i Figur 5.4, kontra beste modellen oppnådd i oppgaven på 0.979 vist i Figur 5.3. Den ikke-styrte lærings-metoden klarer å klassifisere klassen hviler bedre enn beste resultatet fra de tilbakevendende nevrale nettverkene, men dårligere på klassen bevegelse.



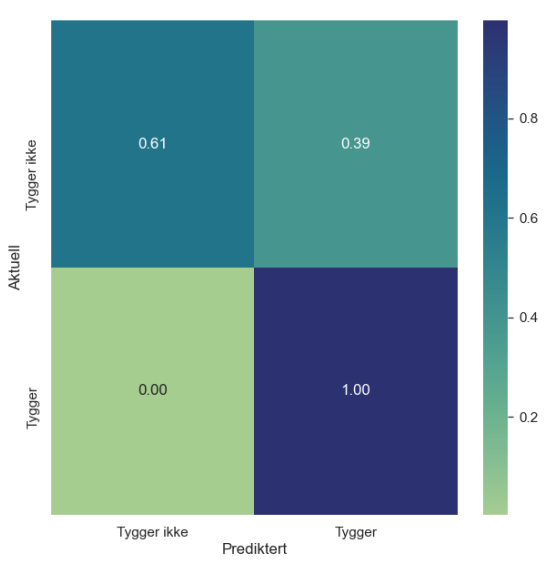
Figur 5.3: Beste resultat for binær bevegelse, GRU med oppdeling 1.



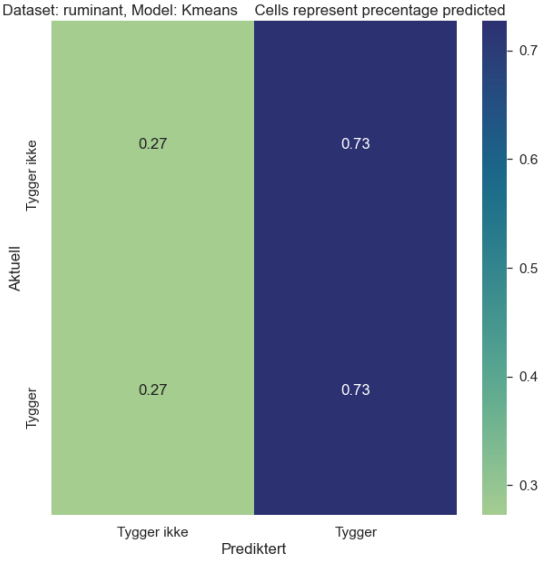
Figur 5.4: Forvirringsmatrise k-means clustering med 2 grupperinger.

5.2.3 Sammenlikning - Drøvtygging

Drøvtygging datasettet med *k-means clustering* ga en F1-score på 0.754 vist i Figur 5.6, kontra beste modellen oppnådd i oppgaven på 0.956 vist i Figur 5.5. Tilbakevendende nevrale nettverk gjør en bedre klassifisering for begge atferdene.



Figur 5.5: Beste resultat for drøvtygging, GRU med oppdeling 2.



Figur 5.6: Forvirringsmatrise k-means clustering med 2 grupperinger, drøvtygging.

5.3 Datagrunnlaget

Datasettene brukt i denne studien er kun fra to kalver over en kort tidsperiode, så nettverkene får ikke nok variasjon til å kunne generalisere bra på ny data fra en annen kalv. Det er også andre faktorer i datagrunnlaget som tilsier lav variasjon. Og en av de er at størrelsen på forsøksområdet er lite, med små terrengvariasjoner. En annen faktor er liten temporal variasjon siden dataene er fra kun to dager i året, og som også knyttes sammen med dårlig variasjon i værtype. Alle disse faktorene er med på å skape små variasjoner i datagrunnlaget, som er viktig for god generalisering.

Klassifisering av atferder med hjelp av video observasjoner er nevnt tidligere som en kilde til feil. For å unngå slike feil er det viktig å i ettertid bruke mye tid på å studere de dannede datasettene før maskinlæringen settes i gang. Å tegne opp grafen for signalene sammen med klasseinndelingen kan for eksempel gi en god pekepinn på om koblingen mellom akselerometeret og observasjonene er gjort korrekt.

5.4 Nettverksmodeller og hyperparameter innstillinger

Det er kun laget maskinlærings modeller med fastsatte antall og type lag, hvor kun parameterne i modellen er endret. Ved å bruke enten bedre maskinvare eller mer tid så er mulig finne enda bedre modeller ved å inkludere antall lag og type lag som egne parametre, man må bare huske at dette betydelig øker antall modellkombinasjoner som må testes.

Antall parametere og parameterkombinasjoner for modellene er i denne studien bevisst valgt basert på tilgjengelig maskinvare spesifikasjoner og tidsbruk. Tid brukt på å trene en modell varierte med antall epochs gjennomgått før earlystop ble aktivert. Man kunne observere store forskjeller i tidsbruk mellom GRU og LSTM modellene, GRU kunne bruke litt over 1 time på 1000 modeller og LSTM opptil 8 timer på 1000 modeller. GRU sammen med konvulsjons lag er en mye enklere modell enn flerlags LSTM.

5.5 Forslag til videre arbeid

Dagens teknologi har gjort det mulig å studere og predikere atferder hos husdyr. Kjennskap til disse atferdene muliggjør kontinuerlig overvåkning av dyrehelsen og bedrer dyre-forvaltningen om man bygger videre på samspillet mellom sensor- og programvareteknologien. Ved å innlemme teknologi- og programvareutviklere kan man utvikle og tilby kommersielle løsninger for de omfattende husdyrindustriene.

Store steg innen husdyrhold er allerede gjort av selskaper som Nofence. Nofence har klart å utvikle et kommersielt produkt basert på et akselerometer og GPS, som fjerner behovet for husdyr som geit, sau og storfe å måtte være inngjerdet ved å introdusere virtuelle gjerder [25]. Framtidens husdyrhold vil uten tvil fortsette å ta i bruk slike teknologier, men man må også fortsette å utvikle metoder som kan muliggjør predikeringer av mer komplekse atferder. Eksempler for dette kan man observere i denne studien hvor det er konkludert med at det kreves mer informasjon og analyse for å muliggjøre predikering av atferder som diing. Framtiden bringer med seg flere metoder for datainnsamling og effektive sensorer, samt muligheter for predikering av stadig mer komplekse atferder. Utviklingen av mindre og mer kraftfulle datamaskiner åpner også opp for at tilbakevendende nevrale nettverk i sanntid kan klassifisere atferder inne i klaven.

Målet i denne studien er ikke å danne en ferdig trent maskinlæringsmodell, men heller å danne og teste en framgangsmåte som kan brukes på ny og forbedret data basert på funnene gjort. Dataprogramvaren brukt og utviklet i denne studien kan med små modifikasjoner videreutvikles til å ta imot nye datasett og andre typer sensorer for nye dataattributter.

Ved å inkludere data fra andre kilder kan man forbedre resultatene på atferds-klassifiseringene. Eksempler på ekstra datakilder som kan brukes til atferds-klassifisering er:

- GPS
- Gyroskop (IMU)
- Værdata og topografi

Det er viktig å poengtere at modellene senere bare kan brukes på samme type data de er trent opp på, som vil si samme attributter.

En sterk anbefaling er å samle inn mer data fra flere kalver, selv om video-observasjons klassifiseringen er mye arbeid så vil en dobling av antall kalver fra 2 til 4 og lengre perioder kunne gi en mye mer robust modell med bedre resultater.

6 Konklusjon

Ut ifra de tre testede datasettene så er det Gated Recurrent Unit (GRU) arkitekturen i kombinasjon med konvolusjonslag og samle-lag som ga de beste resultatene, som samsvarer med fordelen GRU har ovenfor LSTM på små datasett. Ved å ha studert resultater fra modeller med ulike oppdelinger, parametre og nøyaktighet så er det oppdaget hvor viktig ikke bare datagrunnlaget er, men også hvordan datagrunnlaget blir behandlet før maskinlæring. Oppgaven belyser viktigheten ved å studere dataene og effekten av å ha ubalanserte datasett.

Resultatene denne studien viser at det er mulig å trene opp tilbakevendende nevrale nettverks til å predikere enkelte atferder hos kalver med kun akselerometerdata, men sliter med å klassifisere og skille atferder med like signaler. Resultatene viser også at man trenger betydelig mer data for at modellene skal kunne generalisere bedre. Et annet anbefalt alternativ baserer seg på å inkludere data fra andre sensorer, men bare i tilfeller hvor dette er hensiktsmessig. Det anbefales å jobbe videre med denne metoden på nye data i form av et større forsøk. Og ikke bare for kalv, men også på andre husdyr siden metoden kan brukes på andre legemer i bevegelse med få modifikasjoner.

Bruksområdet for klassifisering av atferder med tilbakevendende nevrale nettverk strekker seg mye lengre enn til kun atferds-studie for kalv. Metoden kan potensielt brukes for all bevegelses klassifisering av legemer, samt kunne ekspandere modellene til å kunne klassifisere atferder i sanntid. Oversikt over atferd hos husdyr er med på å skape en mer bærekraftig gårdsdrift om de viktigste atferdene er klassifisert nøyaktig nok til å kunne predikere tilstanden på dyret.

Bibliografi

[1] Vahid Mirjalili Sebastian Raschka. Python Machine Learning - Second Edition: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow. Packt, 2017.

[2] Christopher Olah. Understanding lstm networks. Unknown Journal, 2015.

[3] Simeon Kostadinov. Understanding gru networks. Unknown Journal, 2017.

[4] Yingqi Peng, Naoshi Kondo, Tateshi Fujiura, Tetsuhito Suzuki, Hidetsugu Yoshioka, Erina Itoyama, et al. Classification of multiple cattle behavior patterns using a recurrent neural network with long short-term memory and inertial measurement units. Computers and Electronics in Agriculture, 157:247–253, 2019.

[5] Jamie Barwick, David William Lamb, Robin Dobos, Mitchell Welch, Derek Schneider, and Mark Trotter. Identifying sheep activity from tri-axial acceleration signals using a moving window classification model. Remote Sensing, 12(4):646, 2020.

[6] Harvey J Miller, Somayeh Dodge, Jennifer Miller, and Gil Bohrer. Towards an integrated science of movement: converging research on animal movement ecology and human mobility science. International Journal of Geographical Information Science, 33(5):855–876, 2019.

[7] Eloise S Fogarty, David L Swain, Greg M Cronin, Luis E Moraes, and Mark Trotter. Behaviour classification of extensively grazed sheep using machine learning. Computers and Electronics in Agriculture, 169:105175, 2020.

[8] Somayeh Dodge. A data science framework for movement. Geographical Analysis, 53(1): 92–112, 2021.

[9] Chuan Liao, Patrick E Clark, Mohamed Shibia, and Stephen D DeGloria. Spatiotemporal dynamics of cattle behavior and resource selection patterns on east african rangelands: evidence from gps-tracking. International Journal of Geographical Information Science, 32(7): 1523–1540, 2018.

[10] Daniel Smith, Ashfaqur Rahman, Greg J Bishop-Hurley, James Hills, Sumon Shahriar, David Henry, and Richard Rawnsley. Behavior classification of cows fitted with motion collars: Decomposing multi-class classification into a set of binary problems. Computers and Electronics in Agriculture, 131:40–50, 2016.

[11] L A González, G J Bishop-Hurley, Rebecca N Handcock, and Christopher Crossman. Behavioral classification of data from collars containing motion sensors in grazing cattle. Computers and Electronics in Agriculture, 110:91–102, 2015.

[12] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. Nature, 521 (7553):452–459, 2015.

[13] Lina Zhou, Shimei Pan, Jianwu Wang, and Athanasios V Vasilakos. Machine learning on big data: Opportunities and challenges. Neurocomputing, 237:350–361, 2017.

[14] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In Ijcai, volume 15, pages 3995–4001. Buenos Aires, Argentina, 2015.

[15] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. Journal of Systems Engineering and Electronics, 28(1):162–169, 2017.

[16] Michael Hüsken and Peter Stagge. Recurrent neural networks for time series classification. Neurocomputing, 50:223–235, 2003.

[17] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

[18] Gang Chen. A gentle tutorial of recurrent neural network with error backpropagation. arXiv preprint arXiv:1610.02583, 2016.

[19] Simone Sharma, Sagar. Activation functions in neural networks. Towards Data Science, 6 (12):310–316, 2017.

[20] Sebastian Ruder. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:1609.04747, 2016.

[21] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. arXiv preprint arXiv:2010.16061, 2020.

[22] Raphael Vallat. Pingouin: statistics in python. Journal of Open Source Software, 3(31):1026, 2018.

[23] Wes McKinney and PD Team. Pandas-powerful python data analysis toolkit. Pandas—Powerful Python Data Analysis Toolkit, 1625, 2015.

[24] Shudong Yang, Xueying Yu, and Ying Zhou. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI), pages 98–101. IEEE, 2020.

[25] Emma Brunberg, Rose Bergslid, and Kristin Sørheim. The virtual fencing system nofence-trials 2013. Bioforsk Rapport, 2013.

Tillegger

A Resultater

A.1 Flerklasset bevegelse

Tabell A.1.1: Nøyaktighet for beste modell, flerklasset GRU, oppdeling 1.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.946015	0.880383	0.912020	418.0
Bevegelse	0.894410	0.688995	0.778378	209.0
Beiter	0.781189	0.992570	0.874284	1615.0
Dier	0.125000	0.007812	0.014706	384.0
Snitt/Total	0.720482	0.806550	0.746962	2626.0

Tabell A.1.2: Nøyaktighet for nest beste modell, flerklasset GRU, oppdeling 1.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.966234	0.889952	0.926526	418.0
Bevegelse	0.988636	0.416268	0.585859	209.0
Beiter	0.763145	0.997523	0.864734	1615.0
Dier	0.000000	0.000000	0.000000	384.0
Snitt/Total	0.701824	0.788271	0.725925	2626.0

Tabell A.1.3: Nøyaktighet for beste modell, flerklasset GRU, oppdeling 2.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.767779	0.875828	0.818252	604.0
Bevegelse	0.276923	0.315789	0.295082	57.0
Beiter	0.670732	0.743243	0.705128	74.0
Dier	0.000000	0.000000	0.000000	121.0
Snitt/Total	0.618175	0.703271	0.657971	856.0

Tabell A.1.4: Nøyaktighet for nest beste modell, flerklasset GRU, oppdeling 2.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.767802	0.823920	0.794872	301.0
Bevegelse	0.175439	0.344828	0.232558	29.0
Beiter	0.750000	0.710526	0.729730	38.0
Dier	0.000000	0.000000	0.000000	60.0
Snitt/Total	0.618449	0.665888	0.639557	428.0

Tabell A.1.5: Nøyaktighet for beste modell, flerklasset LSTM, oppdeling 1.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.945876	0.877990	0.910670	418.0
Bevegelse	0.925000	0.360976	0.519298	205.0
Beiter	0.750000	0.999383	0.856916	1621.0
Dier	0.000000	0.000000	0.000000	384.0
Snitt/Total	0.685217	0.784247	0.713918	2628.0

Tabell A.1.6: Nøyaktighet for nest beste modell, flerklasset LSTM, oppdeling 1.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.827128	0.744019	0.783375	418.0
Bevegelse	0.855856	0.463415	0.601266	205.0
Beiter	0.755255	0.997532	0.859649	1621.0
Dier	0.000000	0.000000	0.000000	384.0
Snitt/Total	0.664177	0.769787	0.701751	2628.0

Tabell A.1.7: Nøyaktighet for beste modell, flerklasset LSTM, oppdeling 2.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.738452	0.989550	0.845757	1244.0
Bevegelse	0.000000	0.000000	0.000000	118.0
Beiter	0.871287	0.560510	0.682171	157.0
Dier	0.000000	0.000000	0.000000	250.0
Snitt/Total	0.596623	0.745619	0.655299	1769.0

Tabell A.1.8: Nøyaktighet for nest beste modell, flerklasset LSTM, oppdeling 2.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.746316	0.936495	0.830660	1244.0
Bevegelse	0.000000	0.000000	0.000000	118.0
Beiter	0.604878	0.789809	0.685083	157.0
Dier	0.000000	0.000000	0.000000	250.0
Snitt/Total	0.578510	0.728660	0.644940	1769.0

A.2 Binærklasset bevegelse

Tabell A.2.1: Nøyaktighet for beste modell, binærklasset GRU, oppdeling 1.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.997253	0.872596	0.930769	416.0
Bevegelse	0.976569	0.999548	0.987925	2210.0
Snitt/Total	0.979846	0.979436	0.978871	2626.0

Tabell A.2.2: Nøyaktighet for nest beste modell, binærklasset GRU, oppdeling 1.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.997207	0.858173	0.922481	416.0
Bevegelse	0.973986	0.999548	0.986601	2210.0
Snitt/Total	0.977664	0.977152	0.976443	2626.0

Tabell A.2.3: Nøyaktighet for beste modell, binærklasset GRU, oppdeling 2.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.795222	0.774086	0.784512	301.0
Bevegelse	0.496296	0.527559	0.511450	127.0
Snitt/Total	0.706522	0.700935	0.703487	428.0

Tabell A.2.4: Nøyaktighet for nest beste modell, binærklasset GRU, oppdeling 2.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.784053	0.784053	0.784053	301.0
Bevegelse	0.488189	0.488189	0.488189	127.0
Snitt/Total	0.696262	0.696262	0.696262	428.0

Tabell A.2.5: Nøyaktighet for beste modell, binærklasset LSTM, oppdeling 1.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.785714	0.925743	0.850000	202.0
Bevegelse	0.985479	0.952292	0.968601	1069.0
Snitt/Total	0.953731	0.948072	0.949752	1271.0

Tabell A.2.6: Nøyaktighet for nest beste modell, binærklasset LSTM, oppdeling 1.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.736220	0.925743	0.820175	202.0
Bevegelse	0.985251	0.937325	0.960690	1069.0
Snitt/Total	0.945672	0.935484	0.938358	1271.0

Tabell A.2.7: Nøyaktighet for beste modell, binærklasset LSTM, oppdeling 2.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.749330	0.925497	0.828148	604.0
Bevegelse	0.590909	0.257937	0.359116	252.0
Snitt/Total	0.702692	0.728972	0.690069	856.0

Tabell A.2.8: Nøyaktighet for nest beste modell, binærklasset LSTM, oppdeling 2.

Klasse	Presisjon	Recall	f1	Fordeling
Hviler	0.745407	0.943522	0.832845	301.0
Bevegelse	0.638298	0.236220	0.344828	127.0
Snitt/Total	0.713624	0.733645	0.688036	428.0

A.3 Drøvtygging

Tabell A.3.1: Nøyaktighet for beste modell, drøvtygging GRU, oppdeling 1.

Klasse	Presisjon	Recall	f1	Fordeling
Tygger ikke	0.500000	0.800000	0.615385	5.0
Tygger	0.987179	0.950617	0.968553	81.0
Snitt/Total	0.958855	0.941860	0.948020	86.0

Tabell A.3.2: Nøyaktighet for nest beste modell, drøvtygging GRU, oppdeling 1.

Klasse	Presisjon	Recall	f1	Fordeling
Tygger ikke	0.533333	0.666667	0.592593	12.0
Tygger	0.972973	0.953642	0.963211	151.0
Snitt/Total	0.940607	0.932515	0.935926	163.0

Tabell A.3.3: Nøyaktighet for beste modell, drøvtygging GRU, oppdeling 2.

Klasse	Presisjon	Recall	f1	Fordeling
Tygger ikke	0.937500	0.612245	0.740741	49.0
Tygger	0.961145	0.995763	0.978148	472.0
Snitt/Total	0.958921	0.959693	0.955820	521.0

Tabell A.3.4: Nøyaktighet for nest beste modell, drøvtygging GRU, oppdeling 2.

Klasse	Presisjon	Recall	f1	Fordeling
Tygger ikke	0.823529	0.608696	0.700000	23.0
Tygger	0.963115	0.987395	0.975104	238.0
Snitt/Total	0.950814	0.954023	0.950861	261.0

Tabell A.3.5: Nøyaktighet for beste modell, drøvtygging LSTM, oppdeling 1.

Klasse	Presisjon	Recall	f1	Fordeling
Tygger ikke	0.529412	0.418605	0.467532	43.0
Tygger	0.949393	0.967010	0.958121	485.0
Snitt/Total	0.915190	0.922348	0.918167	528.0

Tabell A.3.6: Nøyaktighet for nest beste modell, drøvtygging LSTM, oppdeling 1.

Klasse	Presisjon	Recall	f1	Fordeling
Tygger ikke	0.420000	0.488372	0.451613	43.0
Tygger	0.953975	0.940206	0.947040	485.0
Snitt/Total	0.910488	0.903409	0.906693	528.0

Tabell A.3.7: Nøyaktighet for beste modell, drøvtygging LSTM, oppdeling 2.

Klasse	Presisjon	Recall	f1	Fordeling
Tygger ikke	0.000000	0.000000	0.000000	8.0
Tygger	0.905882	1.000000	0.950617	77.0
Snitt/Total	0.820623	0.905882	0.861147	85.0

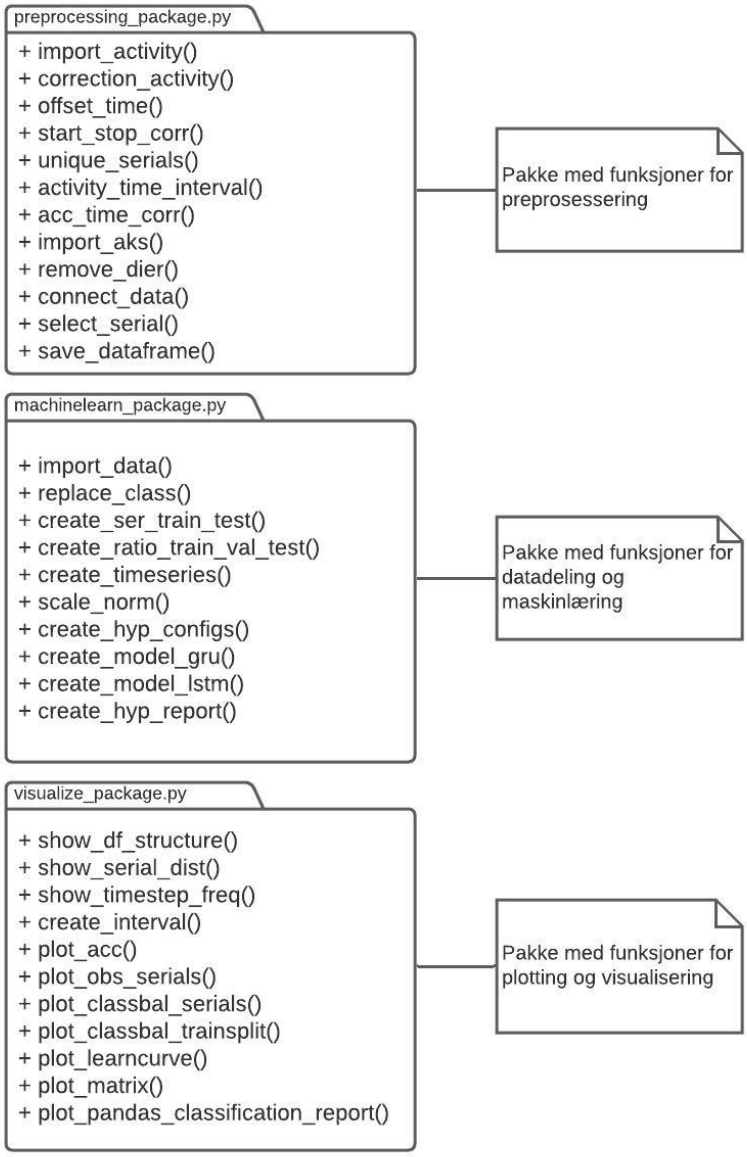
Tabell A.3.8: Nøyaktighet for nest beste modell, drøvtygging LSTM, oppdeling 2.

Klasse	Presisjon	Recall	f1	Fordeling
Tygger ikke	0.000000	0.000000	0.000000	8.0
Tygger	0.905882	1.000000	0.950617	77.0
Snitt/Total	0.820623	0.905882	0.861147	85.0

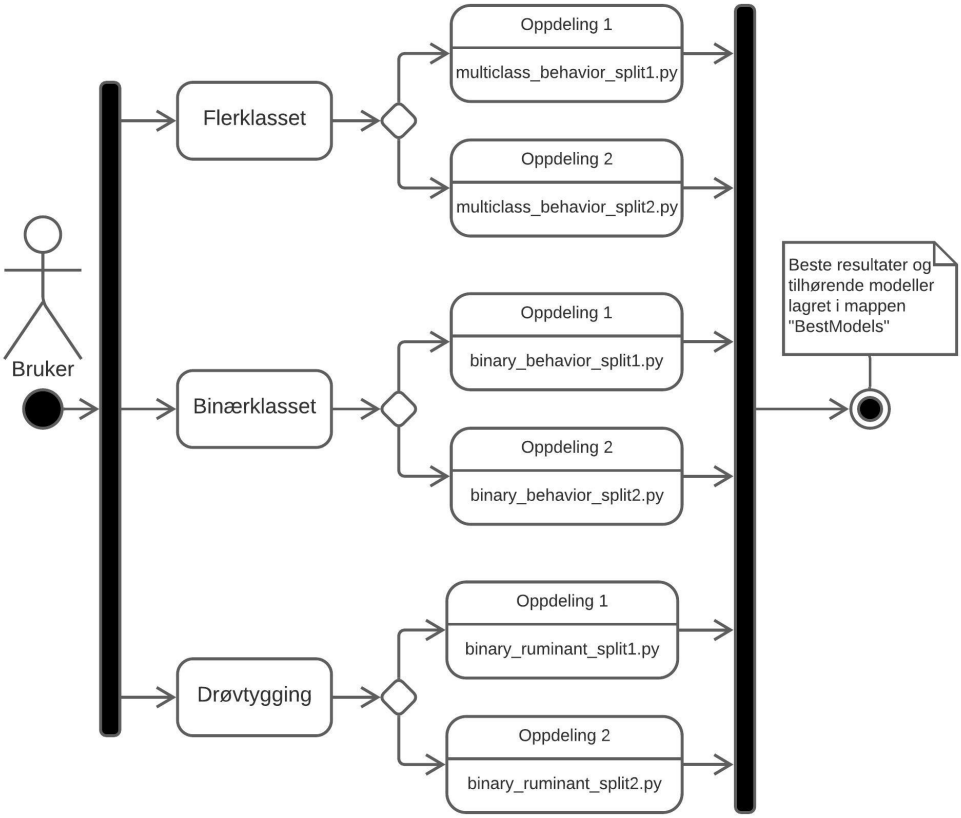
B Python flytskjema

Kode produsert for denne oppgaven er tilgjengelig på GitHub:
`https://github.com/knut-henning/MovementClassification_Thesis`

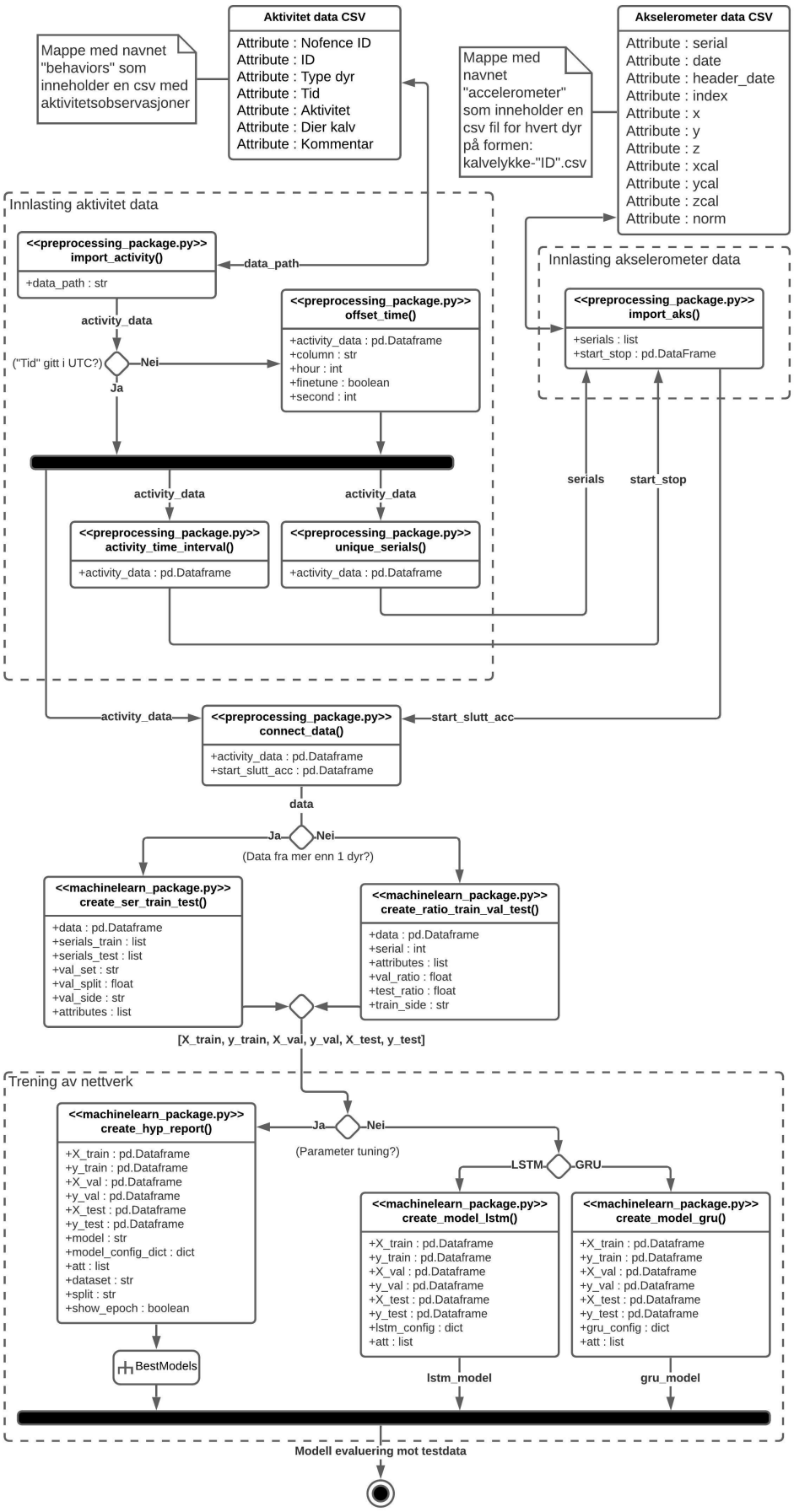
Men for å kunne kjøres kreves tilgang på akselerometerdata fra Nofence eller egne data.



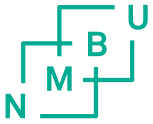
Figur B.1: Genererte program pakker til oppgaven.



Figur B.2: Python filer brukt for å generere resultatene i oppgaven.



Figur B.3: Dataflyt gjennom genererte funksjoner, se funksjons info i Python fil for mer info.



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway