FFI-rapport 2008/00443

Klassifikasjon i SAR-bilder ved bruk av ulike preprosesseringsteknikker og klassifikatorer

Nina Ødegaard

Forsvarets forskningsinstitutt (FFI)

19. desember 2008

FFI-rapport 2008/00443

1059

P: ISBN 978-82-464-1509-3
E: ISBN 978-82-464-1510-9

Emneord

SAR

ATR

Godkjent av

Terje Johnsen	Prosjektleder
Johnny Bardal	Avdelingssjef

Sammendrag

Automatisk målgjenkjenning (ATR) i SAR-bilder kan lette arbeidet til en operatør. En høyoppløselig SAR-sensor vil kunne generere store mengder data, og det er derfor ønskelig å automatisere klassifikasjonsprosessen. For at ATR skal kunne brukes operativt må den være robust nok til at en operatør stoler på den. Dette er den store utfordringen med SAR ATR, og grunnen til at det ikke brukes operativt i noen særlig grad idag. Det er vanskelig å finne algoritmer som er robuste, bl.a. når bakgrunnen endrer seg, objektet er forsøkt skjult etc.

Denne rapporten gir en kort oversikt over fagfeltet, samt en oppsummering av hva vi har sett på av metoder i prosjekt SOBEK. Den vil samtidig si noe om hva som bør gjøres videre i prosjektet på dette området. Mye av det som nevnes er ennå ikke implementert, men vi mener det er nyttig å ha det med for å gi en helhetlig oversikt.

To hovedmåter å tilnærme seg problemet på nevnes, en som baserer seg på innsamlede data og en som baserer seg på modellering av signaturer. Analysene i rapporten bruker den første tilnærmingen, da vi har tilgang til slike data. De forskjellige delene av SAR ATR prosesseringskjeden er beskrevet. Dette omfatter deteksjon, segmentering og annen preprosessering av bildene. Videre presenteres et utvalg av aktuelle egenskaper man kan tenke seg å bruke i en klassifikator, samt metoder for å finne de 'optimale' egenskapene fra et utvalg.

Fordi egenskaper ved målet ofte er avhengige av aspektvinkelen mellom radaren og målet, har vi brukt endel tid på å estimere denne vinkelen. En metode som baserer seg på fusjon av ulike estimater blir beskrevet. Det viser seg i våre tester at det alltid lønner seg å bruke dette samlede estimatet istedenfor noen av de individuelle. En lignende metode demonstreres også for å estimere målenes lengde og bredde. Tre klassifikatorer har blitt implementert, og disse bekrives sammen med resultater av tester. Testene viser et relativt godt resultat med k-nærmeste nabo-metoden. Dette er imidlertid den tregeste metoden. Med minimum feilrate-metoden og neuralt nettverk blir resultatene dårligere, og vi må derfor jobbe videre med å finne bedre egenskaper som kan skille mellom de ulike klassene i treningssettet. Andre klassifikatorer som vil bli forsøkt implementert etterhvert nevnes også.

For å kunne sammenligne en klassifikator med andre er det viktig å gi et kvantitativt mål på ytelsen. Det er ulike måter å gjøre dette på i ATR-miljøene. Vi har valgt å presentere og bruke en evalueringsmåte som vi mener gir et helhetlig bilde av ytelsen.

English summary

Automatic target recognition (ATR) in SAR imagery kan reduce the workload for an analyst. A high resolution SAR sensor can produce large amounts of data, and it is therefore desirable to automate the classification process. In order to use ATR in the field it has to be robust enough to make the analyst trust it. This is the main challenge with SAR ATR, and the reason why it is not used to any extent in the field. It is difficult to find algorithms that are robust, for example when the background changes or if the object is hidden.

This report will give a short overview of SAR ATR, together with a presentation of what has been done on this subject in project SOBEK. It will also indicate what should be done in the future. A lot of what is mentioned is still not implemented, but we think it is instructive to include it to get a more complete overview.

Two main approaches to solve the problem will be described, one that is based on recorded data and one that is based on modelled signatures. The analyses in this report will use the first approach, since we have access to such data. The different parts of the SAR ATR processing chain will be described. This includes detection, segmentation and other preprocessing of the images. Then a selection of features that may be useful in a classifier is presented, along with methods to find the 'optimal' features from such a selection.

As features of a target often depend on the aspect angle between the sensor and the target, we have tried to estimate this angle. A method based on fusion of different estimates will be described. From our tests we see that it is always feasible to use this combined estimate over the individual estimates. A similar method will be demonstrated to estimate the length and the width of the targets. Three classifiers have been implemented, and they will be described along with testresults. The tests show a relatively good result for the k-nearest neighbour method. However, this is the slowest method. The results for the minimum error rate method and the neural network are not so good. We therefore have to work harder to find better features that can separate the different classes in the training set. Other classifiers that will be implemented at a later time are also mentioned.

To compare one classifier with others it is important to be able to give a quantitative measure of the performance. Different measures have been used in different ATR communities. We have chosen to present and use one evaluation method here that we feel gives a complete picture of the performance.

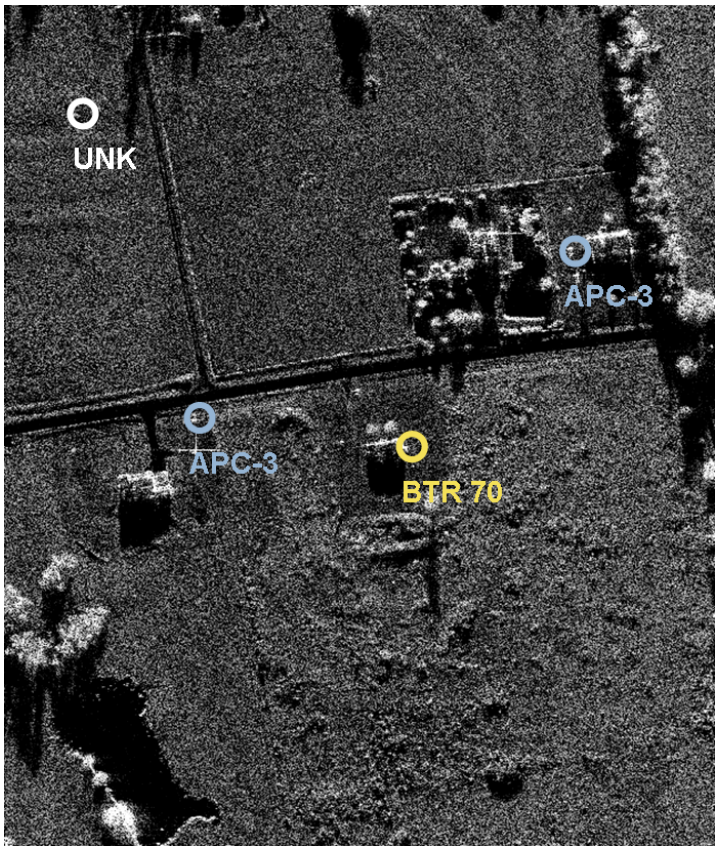
Innhold

1 Innledning

Automatisk målgjenkjenning (ATR) i SAR-bilder kan lette arbeidet til en operatør. Fordi SAR-bilder avbilder scenen på en annen måte enn et fotografi kan det være vanskelig for en operatør å identifisere forskjellige objekter i bildene. Mye trening må ofte til før man blir en god tyder av SAR-bilder. En høyoppløselig SAR-sensor vil kunne generere store mengder data, og det er derfor ønskelig å automatisere denne klassifikasjonsprosessen. Å kunne lage et produkt som i figur 1.1 er ønskelig. Her får man inn et stort SAR-bilde som kan inneholde mange potensielle mål. En algoritme vil kunne detektere de objektene som ser interessante ut og videre klassifisere dem til de forskjellige klassene i treningssettet eller avgjøre at de tilhører en ukjent klasse. For at ATR skal kunne brukes operativt må den være robust nok til at en operatør vil stole på den. Dette er den store utfordringen med SAR ATR, og grunnen til at det ikke brukes operativt i noen særlig grad idag. Det er vanskelig å finne algoritmer som virker når bakgrunnen forandrer seg, objektet er forsøkt skjult etc.

Denne rapporten gir en oversikt over fagfeltet, samt en oppsummering av hva vi har sett på av metoder i prosjekt SOBEK. Den vil samtidig si noe om hva vi kommer til å gjøre videre. Klassifikasjonsprosessen kan deles opp i flere steg. Måter å løse oppgavene i de forskjellige stegene på vil bli beskrevet, men mye av dette har vi ikke implementert ennå. I prosjektet skal vi jobbe mer med disse temaene, så rapporten må leses som en foreløpig tilstandsrapport. Rapporten omhandler bare klassifikasjon av kjøretøy, men metodene er generelle og vil kunne anvendes på andre objekter man ønsker å klassifisere.

I kapittel 2 presenteres arbeid som har blitt utført av andre, men som denne rapporten i stor grad baserer seg på. Kapittel 3 ser på valg av treningsdata, samt presenterer noen datasett vi har tilgang til. En oversikt over SAR ATR prosesseringskjeden blir presentert i kapittel 4. Videre blir de ulike stegene i denne kjeden forklart nøyere i kapittel 5 om deteksjon, kapittel 6 om preprosessering og kapittel 7 om segmentering. Deretter gis en oversikt over ulike egenskaper man kan tenke seg å bruke i en klassifikator i kapittel 8. Egenskaper ved målet som aspektvinkel, lengde og bredde har vist seg å være viktige å estimere. Metoder for å gjøre dette presenteres i kapittel 9. I kapittel 10 ser vi på endel klassifikatorer som har blitt implementert i prosjekt SOBEK. Standard måter å måle klassifikatorenes ytelse på beskrives i kapittel 11. Resultaene fra tester vi har gjort presenteres i kapittel 12, og til slutt gir vi en konklusjon i kapittel 13.



Figur 1.1: Ønskelig produkt av klassifikasjonsprosessen. Her er interessante mål detektert og klassifisert.

2 Tidligere arbeid

Flere forskningmiljøer har jobbet mye med SAR ATR. Likevel er det ingen som har funnet en optimal og robust løsning på problemet. I det operative brukes ATR foreløpig kun på forsøksstadiet. Her kreves det interaksjon fra operatøren for å få brukbare resultater. Målet er å finne en ATR operatøren stoler på.

2.1 NATO SET-111

NATO Research and Technology Organisation (RTO) har nedsatt en arbeidsgruppe (SET-111, tidligere SET-53) med tittelen 'Effective solutions for the radar ATR multi parameter problem'. Gruppen består av representanter for ATR forskningsmiljøer fra 12 NATO-land, inkludert Norge representert ved FFI. Gruppens mål har ikke vært å finne den optimale løsningen på ATR-problemet, men istedet å koordinere ATR-arbeidet og oppmuntre til samarbeid og utveksling av informasjon mellom deltagerne. SET-53 har laget en sluttrapport som prøver å gi en oversikt over de enkelte landenes arbeider innen ATR [1]. Denne rapporten gir en god oversikt over de mange mulige måtene man kan forsøke å løse ATR-problemet på. Prosjekt SOBEK har i stor grad brukt denne rapporten for å finne metoder å teste, samt for å få en oversikt over hva som er blitt gjort tidligere på dette området.

2.2 Annet arbeid

På FFI er det laget en oversikt over utfordringer på dette fagområdet som tar for seg hva som skal til for å gjøre ATR i SAR-/ISAR-bilder. Dette inngår i en større rapport som omhandler SAR, ISAR og MTI mot overflatemål [2]. I tillegg finnes det en mengde andre artikler og bøker om emnet, noen av dem vil bli nevnt videre i rapporten.

3 Datasett

3.1 Målte eller simulerte data?

Det er to hovedmåter man kan nærme seg ATR-problemet på. Man kan enten basere seg på modeller/simuleringer av radarsignaturer fra forskjellige mål, eller man kan gjøre virkelige målinger av disse. Det er både fordeler og ulemper ved begge metodene. I den første metoden lager man en CAD-modell av det aktuelle objektet, og gjør deretter en elektromagnetisk

spredningsprediksjon. Men man kan ikke være helt sikker på at man får simulert mål og bakgrunn slik de er i virkeligheten. Det kan likevel vise seg at simuleringen er god nok for et ATR-system. Det er mange som undersøker hvor detaljerte slike simuleringer må være for å være gode nok. Fordelen med metoden er at det er lett å generere nye treningsdata. Vi har fått tak i programvare for å simulere SAR-bilder. Programmet heter MOCEM og er utviklet av CELAR/DGA i Frankrike [3]. Her benytter man en CAD-modell av objektet, bestemmer radarparametre, og får ut et SAR-bilde. Dette programmet skal vi jobbe mer med senere i prosjektet.

I den andre metoden bruker man korrekte data, men det er altfor dyrt og tidkrevende å samle inn nok data. Man er avhengig av å ha data fra alle interessante mål fra alle mulige aspekt- og elevasjonsvinkler og helst i forskjellige bakgrunner. Dette blir fort en uoverkommelig oppgave. Mange underøker om det kanskje ikke er nødvendig å dekke alle mulige avbildningsgeometrier, men om det finnes noen permanente spredere som holder seg stabile over gitte vinkelintervaller. Om man ikke har nok variasjon i treningssettet er det fort gjort å spesialtilpasse klassifikatoren til akkurat det treningssettet man jobber med. Ytelsen vil da bli god på akkurat dette settet, men så fort man skal klassifisere nye data vil man sannsynligvis få problemer. Prosjekt SOBEK har fått tilgang til noen treningssett med SAR-data som vil bli beskrevet i delkapitlene 3.3, 3.4 og 3.5.

3.2 Komplekse bilder eller intensitetsbilder?

Med komplekse bilder kan man gjøre en del prosessering som ikke er mulig i intensitetsbilder. Dette gjelder f.eks superoppløsning og uttrekking av de mest stabile sprederne, omtalt i delkapittel 6.2, og skyggeforbedring som omtales i delkapittel 6.3. Enkelte, bl.a. [4], mener at det ikke er godt nok å bare se på intensitetsbildet, men at man også må ha med fasen for å kunne klassifisere objekter i SAR-bilder. Så langt i prosjektet har vi konsentert oss om intensitetsbildene, men skal etterhvert se mer på de komplekse bildene.

3.3 MSTAR

The Moving and Stationary Target Acquisition and Recognition program (MSTAR) ble startet av The United States Defence Advanced Research Project Agency (DARPA), og en del av deres innsamlede data har blitt gjort tilgjengelig. På grunn av dette er det mange som har testet ATR-systemer på dette datasettet. Settet består av 3671 komplekse SAR-bilder til trening og 3203 til testing. Bildene kommer fra 10 forskjellige klasser av kjøretøy. Oppløsningen er på ca. 30 cm. Trenings- og testdataene ble samlet inn med litt forskjellige elevasjonsvinkler for å kunne teste om en gitt klassifikator vil være robust nok til å takle

endrede forhold. Figur 3.1 viser eksempler på SAR-bilder fra de 10 forskjellige klassene i settet. Det er MSTAR-datasettet som er brukt i testene i denne rapporten.

3.4 QinetiQ

QinetiQ har laget en samling SAR ATR-data for NATO SET-53. Dette er full-polarimetriske X-bånd-data av 9 forskjellige klasser av mål samlet inn med Enhanced Surveillance Radar (ESR). Vi får desverre ikke vite hva slags mål dette er, bortsett fra at målene A, B, D og G er militære kjøretøy, E og F er sivile kjøretøy og C, H og I er narre-objekter. Spotlight mode ble brukt og målene ble avbildet i forskjellige aspektvinkler fra 0 til 360 grader. Kalibreringsreflektorer ble plasser i scenen og har blitt brukt til å gjøre polarimetrisk kalibrering av bildene. Elevasjonsvinkelen var 6°, noe som har resultert i lange skygger. QinetiQ-dataene ser ut til å inneholde noe mer støy enn MSTAR-dataene, noe som kan gjøre klassifiseringen vanskeligere. En viktig del av utviklingen er å finne ut hvor mye økt støynivå har å si for klassifikasjonen. Da vi ikke har fasiten på typen mål som finnes i dette datasettet har vi foreløpig ikke sett noe på det. Vi vil bruke det senere som ukjente objekter for å forvirre klassifikatorene.

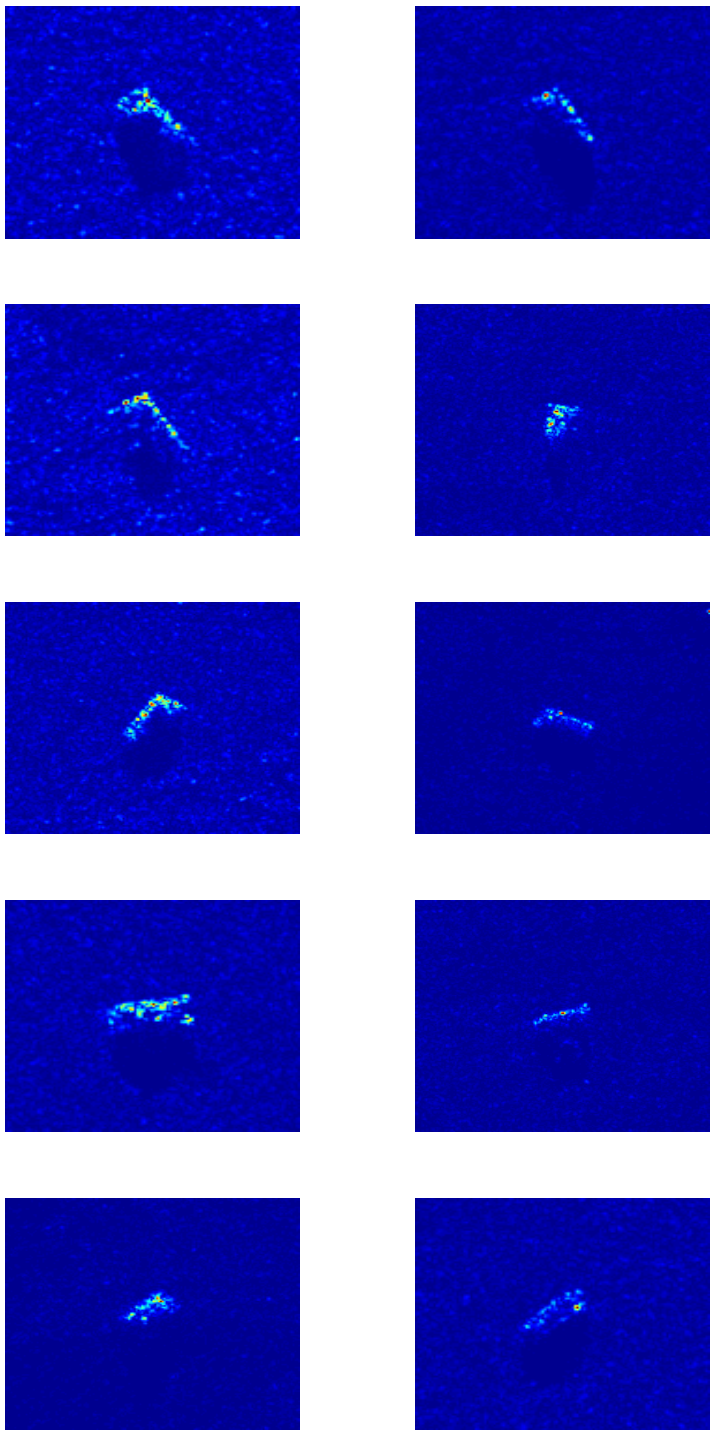
3.5 Sandia

Sandia National Laboratories har lagt en del komplekse SAR-bilder tilgjengelig for alle på nett. Disse dataene ble samlet inn med radaren miniSAR, en Ku-bånd-radar, og oppløsningen er på rundt 10 cm. Det er en del objekter i bildene, f.eks fly, biler og bygninger. Disse objektene kan man bruke som ukjente objekter for å forvirre en klassifikator.

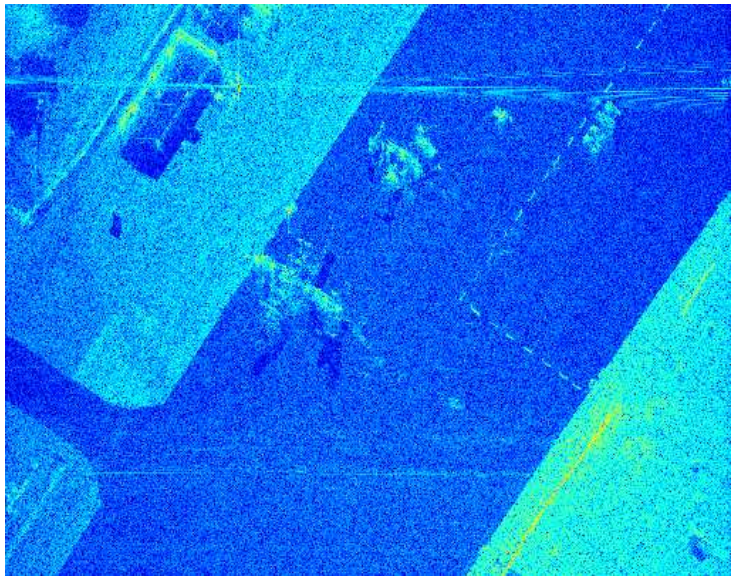
4 SAR ATR prosesseringskjeden

Det er mange steg man må gjennom for å kommme frem til en endelig klassifikasjon av et objekt i et SAR-bilde. Man kan kalle dette for hele ATR prosesseringskjeden. I figur 4.1 ser vi en illustrasjon av de ulike stegene i denne kjeden. Utfordringene i hvert steg kan løses på mange forskjellige måter.

- Man kan tenke seg at vi starter med et SAR-bilde av en stor scene og vil finne de interessante objektene det inneholder automatisk. Kanskje er det mange objekter i bildet, og det kan være vanskelig å se objektene fordi de kan ha havnet i skyggen fra andre



Figur 3.1: Eksempler på de 10 forskjellige klassene av mål i MSTAR-datasettet



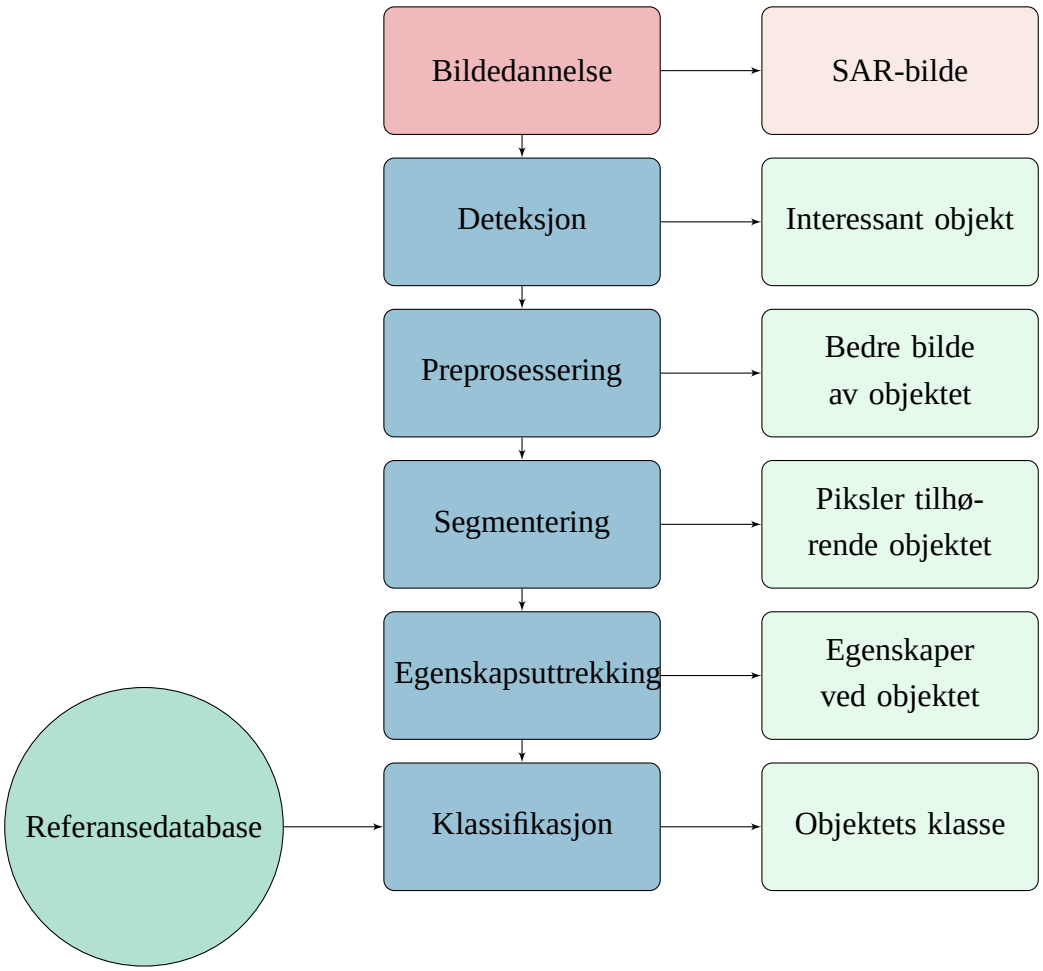
Figur 3.2: SAR-bilde fra Sandia

objekter, fordi bare deler av objektet reflekterer ved den gitte aspektvinkelen, eller fordi noen har forsøkt å skjule det. En algoritme vil kunne finne interessante objekter i bildet basert på sannsynlighetsfordelinger for bakgrunnens og målenes intensitet. Dette steget kalles stort sett bare for deteksjon. Man kan detektere kun objekter eller søke etter objekter sammen med skygge.

- Etter deteksjonen sitter man igjen med et mindre bilde som inneholder kun det aktuelle objektet samt litt bakgrunn. For å gjøre bildet mer egnet for klassifikasjon kan det preprosesseres med bildeforbedringsteknikker, som f.eks superoppløsning, støyfjerning, skyggeforbedring etc. Dette gjøres for at objektet skal komme klarere frem i bildet.

- Deretter må man segmentere ut pikslene som hører til objektet og evt. skyggen, og kvitte seg med bakgrunnen. I dette steget kan man også gjøre en første diskriminering mellom menneskeskapte interessante objekter og uinteressante objekter. Sistnevnte kan f.eks være trær og busker som lett kan plukkes opp som interessant av en deteksjonsalgoritme.

- Så må man trekke ut de egenskapene ved pikslene som kan diskriminere mellom ulike klasser og bruke dem videre i en klassifikasjonsalgoritme. Å finne disse egenskapene er en stor utfordring, for man vil finne egenskaper som er relativt konstante over intervaller av aspektvinkler og som samtidig er unike for hver enkelt klasse. En viktig parameter videre er derfor objektets aspektvinkel i forhold til radaren. Denne er normalt sett ukjent og må estimeres.



Figur 4.1: Figuren viser SAR ATR prosesseringskjeden. Her er boksene i midten operasjoner som utføres på dataene, mens boksene til høyre viser produktene som kommer ut av hver operasjon. Vi antar videre i rapporten at bildedannelsen allerede er gjort, slik at vi starter prosessen med et SAR-bilde.

- Når man så sitter igjen med de egenskapene man tror en klassifikator kan bruke, velger man den klassifikatoren man vil ha. Her finnes det mange å velge mellom.
- For å kunne gi en vurdering av hvor god klassifikasjonen er må vi innom et evalueringssteg. En klassifikasjonsalgoritme må samtidig som den klassifiserer objektene til riktig klasse ikke gi falske alarmer.

Det er også forskjellige måter å definere klassifikasjon på. Det kommer an på hva klassifikatoren skal brukes til. For noen formål vil det være nok å skille mellom sivile og militære kjøretøyer, andre ganger vil man ha klassen militært kjøretøy. Disse klassene kan igjen deles inn i større og mindre subklasser. Enkelte ganger ønsker man å identifisere hvert enkelt kjøretøy, kanskje for å finne ut om det er fiendtlig eller ikke. Vi har valgt å finne klassen kjøretøy. Dette er også det som er gjort i MSTAR-datasettet.

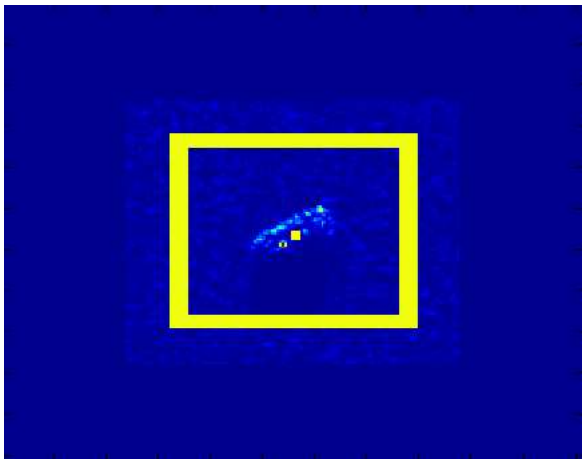
I de følgende kapitlene skal vi se nærmere på de forskjellige stegene i prosesseringskjeden.

5 Deteksjon

Når man får presentert et tilfeldig SAR-bilde er første steg å plukke ut objekter man ønsker å analysere videre. Dette kan være vanskelig fordi bildet kan inneholde mange mål og varierende bakgrunn. Mange mål kan i tillegg være gjemt i landskapet med vilje.

En mye brukt klasse av deteksjonsalgoritmer er de såkalte Constant False Alarm Rate-metodene (CFAR). Det finnes mange varianter av disse, og de retter seg mot alle typer radardata, ikke bare SAR. De prøver alle å holde sannsynligheten for falsk alarm på et konstant nivå, basert på en antatt underliggende fordeling av bakgrunnsclutteret. Dette kan være en gyldig antakelse hvis bakgrunnen er relativt konstant. I et tilfeldig SAR-bilde er det ofte ikke tilfelle. Mange har jobbet med å finne de beste distrubusjonene for å beskrive bakgrunnen i et SAR-bilde, se f.eks. [5] og [6]. Weibull- og K-distrubusjonene blir ofte anbefalt.

Figur 5.1 viser en illustrasjon av filter-oppsettet som brukes for å implementere CFAR. Det ytterste gule området kalles ofte en ring. Distribusjonenes parametre estimeres ut ifra testpikslene i ringen og testes mot pikslene i midten. Både tykkelsen på ringen og antall piksler i ringen kan velges. Antall piksler i midten er også valgfritt, men bør være en funksjon av forventet størrelse på målet. Generelt bør den ytre ringen velges så stor at energi fra et eventuelt mål ikke vil lekke ut til de ytre pikslene. Hvis det er usannsynlig at pikslene i midten kommer fra samme fordeling som pikslene i den ytre ringen blir området plukket



Figur 5.1: Filteroppsett brukt i CFAR. Både størrelsen på ringen og området i midten kan varieres.

ut som et interessant objekt. I referansene [7] and [8] beskrives hvordan man kan estimere parametrene til Weibull-distribusjonen fra testpikslene. I tillegg til å ha en distribusjon for bakgrunnen kan man også velge å ha en for målene. Swerling-modeller velges ofte for dette formålet. Se [5, kapittel 10] for mer generell informasjon om CFAR-deteksjon.

Skygge kan også brukes i deteksjonen. Hvis man observererer skygge i nærheten av et detektert objekt kan det gjøre algorimene sikrere på at det virkelig er et interessant objekt som er plukket ut. Skyggen kan enten detekteres direkte ved hjelp av CFAR på samme måte som man detekterer et objekt, eller man kan sjekke om det rundt et detektert objekt er noe som ligner skygge. I amplitude-bilder vil skyggen ha jevnt lavere pikselverdi enn bakgrunn og mål. Finner man et område av en viss utstrekning med jevnt lave verdier i nærheten av målet kan man regne med at dette er skygge.

I MSTAR-datasettet som er brukt i denne rapporten er objektene allerede detektert, og hvert bilde inneholder kun et objeket samt litt bakgrunn.

6 Preprosessering

Preprosessering av bildene kan gjøre deler av klassifikasjonsprosessen lettere ved at enkelte egenskaper ved bildet forbedres. I dette kapittelet vil tre former for preprosessering nevnes.

6.1 Speckle-reduksjon

Speckle (eller multiplikativ støy) kan fjernes ved multi-looking eller ved hjelp av speckle-filtre. Siden speckle er en sum av bidrag fra mange spredere i en oppløsningscelle, er intensiteten i et piksel med speckle i teorien tilfeldig. Flere look blir da et lavpass-filter, noe som reduserer speckle. Ulempen med dette er at oppløsningen i bildet blir noe dårligere. Multi-looking krever komplekse data. Det er også utviklet flere ulike speckle-filtre som opererer på intensitetsdata. De aller fleste er adaptive og benytter seg av estimater av forventningsverdi og standardavvik fra et område i bildet som antas å være konstant.

Speckle-reduksjon kan gjøre segmenteringen enklere ved at objektet skiller seg klarere ut fra bakgrunnen som blir jevnere.

6.2 Super-oppløsning

Flere metoder har blitt utviklet for å bedre oppløsningen i et allerede eksisterende SAR-bilde. [9], [10] og [11] beskriver flere slike spektrale estimeringsmetoder. De kan også brukes til å trekke ut de mest stabile sprederne fra SAR-bilder, dvs. skille støy fra virkelige spredere på målet. Dette er egenskaper som kan brukes videre i klassifikasjonen. For å kunne bruke disse metodene er man imidlertid avhengig av å ha komplekse bilder.

6.3 Skyggeforbedring

Hvis skyggen av et objekt i et SAR-bilde er veldefinert og gjenspeiler objektets form kan den brukes i klassifikasjonen. [12] har presentert en ny måte å forbedre skyggen i SAR-bilder på. Med forbedring mener de å maksimere kontrasten mellom skygge og bakgrunn slik at man lettere kan se objektets siluett. Metoden kalles Fixed Focus Shadow Enhancement (FFSE). Fordi radaren beveger seg i løpet av integrasjonstida vil også objektets skygge bevege seg. Metoden kompenserer for denne bevegelsen og oppnår dermed skarpere skygge. Komplekse bilder er nødvendig.

7 Segmentering

Når et bilde skal segmenteres antar vi at målet allerde har blitt detektert og at bildet bare inneholder det aktuelle målet og litt bakgrunn. Hvis det er mulig bør området rundt det

målet være så homogent som mulig. Etter segmentering vil man forhåpentligvis bare sitte igjen med piksler som hører til enten målet eller målets skygge for videre analyse.

På grunn av speckle og varierende tilbakespredning fra målet vil en fast terskel for å skille mellom mål og bakgrunn ikke fungere. Den må være adaptiv. I datasett som f.eks MSTAR og QinetiQ der dataene har blitt samlet inn i et kontrollert miljø, vil det være enklere å segmentere enn i data fra en tilfeldig innsamling. Her er bakgrunnen relativt homogen og målene er ikke skjult på noen måte.

7.1 Segmentering av målet

For å segmentere målet kan man bruke det samme filter-oppsettet som ved deteksjon, men med kun ett piksel i midten. I praksis vil man da teste om pikselet i midten overstiger en terskel som avhenger av testpikslenes verdi. Ved å velge denne fremgangsmåten kan man komme til å få med piksler som ikke hører til objektet, men som likevel har høye verdier. Dette kan man komme rundt ved å gruppere pikslene. Da ser man på innbyrdes avstander mellom de utplukkede pikslene. Ved å se på massesenteret i de segmenterte pikslene kan man gå ut en viss avstand i hver retning og si at piksler utenfor dette området er feilsegmentert. En annen måte å gjøre dette på er å søke i en gitt avstand i området rundt hvert piksel for å finne nabo-piksler. Hvis pikselet ikke har noen nabo-piksler i rimelig avstand, kan man anta at det ble plukket ut ved en feiltakelse, og likevel ikke er en del av målet. Denne avstanden vil være en funksjon av forventet størrelse på de målene man leter etter.

Hvis man vil ha en jevnere form på objektet kan det videre være lurt å finne den konvekse omslutningen til pikslene.

7.2 Segmentering av skyggen

Piksler som tilhører objektets skygge kan også være nyttige for klassifikasjonen, spesielt i de projeksjoner hvor skyggen er veldefinert og gjenspeiler formen til objektet. Fra mange aspektvinkler vil ikke skyggen gi ekstra informasjon, men heller forvirre klassifikatoren. Derfor må man være forsiktig når man bruker skyggen på denne måten. Istedenfor å bruke ring-oppsettet er det her mulig å bare velge ut de pikslene med lavest verdi og se om de danner et sammenhengende område av en viss størrelse. Man må også sjekke om skyggen faktisk tilhører målet. For å få en skarpere skygge før segmentering kan skyggeforbedringsteknikker brukes, se delkapittel 6.3.

7.3 Diskriminering

Etter at man har funnet de pikslene man mener hører til målet og eventuelt skyggen, kan man gjøre en forhåndsinndeling i to klasser: menneskeskapt interessant mål eller naturlige objekter som f.eks. trær og busker. De sistnevnte kan også bli plukket opp av en deteksjonsalgoritme da de har et begrenset omfang og ofte skiller seg ut fra bakgrunnen. Til dette kan man bruke objektets form og størrelse, pikselverdier eller annen informsjon man har om målene man regner med å finne. Det er mulig å hoppe over dette steget og heller inkorporere diskrimineringen i selve klassifikatoren ved at man kan klassifisere objekter til en ukjent-klasse. Fordelen med å gjøre en diskriminering på forhånd er at det ikke nødvendigvis er de samme egenskapene som skiller godt mellom de ulike klassene i treningssettet som diskriminerer mellom interessant og uinteressant objekt. Om man diskriminerer på forhånd må man likevel ha med en ukjent-klasse i klassifikatoren, for man kan møte på kjøretøy av andre typer enn de man har i treningssettet.

7.4 Kvalitetssjekk av segmenteringen

For å finne den 'optimale' måten å segmentere på er det nødvendig å gjøre tester med ulike metoder og parametre. Disse parametrene kan være størrelsen på filteret, hvordan man velger ut terskelen som en funksjon av testpikslenes verdier samt metoder for å gruppere pikslene og vurdere hvorvidt man har en skygge som kan brukes.

I store datasett er det ikke hensiktsmessig å se på hvert enkelt resultat av de ulike segmenteringene. Likevel trenger man et mål på hvor godt den aktuelle segmenteringen fungerte. I MSTAR-datasettet er det mulig å finne et slikt kvalitetsmål. Her vet vi at det bare er et mål i hvert bilde, og vi vet hvor stort dette målet er. Dermed kan man regne ut hvor mange piksler målet i teorien skal dekke. Fordi tilbakespredt energi er sterkt avhengig av aspektvinkel kan ikke dette predikeres med sikkerhet. Men man kan sjekke at antallet piksler som ble plukket ut av segmenteringsalgoritmen ligger innenfor et intervall av forventede verdier. I andre datasett der man ikke får vite hvilket mål man ser på eller hvor stort dette målet er, vil det være vanskeligere å avgjøre om segmenteringen var god uten å se på de resulterende bildene.

Dette kvalitetsmålet er foreløpig ikke testet i noen særlig grad i prosjekt SOBEK, men planen er å lagre resultatene i en database for sammenligning. Her vil det bli gjort RMS-analyser av forventet antall piksler sammenlignet med antall utplukkede piksler og det samme for den konvekse omslutningen. En annen måte å evaluere segmenteringene på kan være å kjøre samme klassifikasjonsalgoritmer på data med ulike segmenteringer og se hvilke som gir gode resultater. Dette blir imidlertid en stor jobb, da det er veldig mange parametre, både

i segmenteringen og i klassifikasjonen, som kan varieres. En tredje måte kan være å se på hvor godt vi klarer å estimere aspektvinkel, lengde og bredde med de forskjellige segmenteringsparametrene. En god segmentering vil sannsynligvis også gi gode estimater av disse størrelsene.

8 Egenskaper for klassifikasjon

Generell mønstergjenkjenningsteori sier at i løpet av klassifikasjonsprosessen vil det skje en reduksjon av datamengden. Vanligvis vil man velge ut noen egenskaper som kan diskriminere mellom klassene i treningssettet, og resten av dataene vil bli kastet. Å finne slike egenskaper kan være vanskelig, for man kan ikke nødvendigvis se noe klart mønster i alle de mulige kombinasjonene. Det er mulig å bruke alle inngangsdataene som egenskaper, i dette tilfellet vil det si hele SAR-bildet. Da slipper man helt unna prosessen med å velge ut de mest egnede egenskapene, men til gjengjeld blir prosesseringstiden lang og det kreves stor lagringsplass. I andre enden av skalaen finner vi egenskapsvektorer der individuelle egenskaper ved objektene trekkes ut av bildet. Disse kan også bli store, men man prøver å bare ta med seg den informasjonen som er relevant for klassifikasjonen. Flere egenskaper kan også kombineres og tranformeres på ulike måter for å skape nye egenskaper som forhåpentligvis diskriminerer bedre mellom klassene. Egenskaper som hver for seg ikke ser ut til å kunne diskriminere mellom klassene kan ofte fungere bra når de kombineres eller transformeres på ulike måter. De forskjellige typene av egenskaper diskuteres videre i dette kapittelet.

8.1 SAR-bilde

Man kan kjøre korrelasjon direkte på SAR-bildene og finne ut hvilke som er mest like, såkalt 'template matching'. Dette tar imidlertid veldig lang tid og man må lagre mye data. Man må ha bilder av mål fra alle klassene med mange forskjellige avbildningsgeometrier.

8.2 High resolution range profiler (HRR)

HRR-profiler fås direkte fra ISAR-målinger der radaren står i ro mens objektet beveger seg. De kan også trekkes ut av et SAR-bilde, men det krever litt prosessering. De resulterende profilene kan summeres slik at man står igjen med en profil per bilde. På dette sparer man tid og lagringsplass i forhold til å bruke hele bildet, men mister informasjon. Profilene kan så brukes som egenskapsvektorer i en klassifikator. Siden profilene avhenger sterkt

av aspektvinkelen, vil man i en klassifikator bare sammenligne profiler fra den samme vinkelsektoren som profilen man analyserer. Man kan definere en sektor til å være det vinkelspennet der tilbakespredningen ikke endrer seg nevneverdig. Mange har analysert profilenes avhengighet av aspektvinkelen, bl. a. [13]. [14] har brukt MSTAR-datasettet til å lage slike HRR-profiler og videre klassifisert dem med en såkalt skjult Markov-modell.

8.3 Enkeltstående egenskaper

Vi er ute etter egenskaper som er unike for hver klasse og som samtidig ikke endrer seg nevneverdig avhengig av tid eller avbildningsgeometri. Egenskaper kan f.eks være statistikk i bildet, antall piksler i objektet, form på objektet, sterke spredere etc. Det er en nesten uendelig mengde egenskaper man kan trekke ut av et bilde, og disse kan kombineres på ulike måter for å lage egenskapsvektorer. Antallet kombinasjoner er er så høyt at de ikke kan analyseres direkte, og mange har utarbeidet forskjellige fremgangsmåter for å plukke ut de 'optimale' egenskapene. Se f.eks [15, kapittel 10]. Man vil ikke ha for mange egenskaper i en klassifikator, og det er derfor viktig å finne robuste egenskaper, dvs. egenskaper som vil diskriminere mellom klassene selv om aspekt- og elevasjonsvinkelene endres og selv om målet skulle vært påmontert utstyr eller ha bevegelige deler. Enkelte egenskaper vil lett kunne plukkes ut fra et bilde, mens andre fremkommer kanskje bare ved å transformere det opprinnelige bildet på en eller annen måte.

Det er enkeltstående egenskaper som er blitt testet i denne rapporten. En del aktuelle egenskaper beskrives i de følgende underpunktene.

8.3.1 Geometriske egenskaper

Geometriske egenskaper er de som sier noe om formen til objektet. Dette kan f.eks være parametrene til den ellipsen som best omslutter de segmenterte pikslene, parametrene til rektangelet som best omslutter de segmenterte pikslene, den konvekse omslutningen, massesenteret, antall piksler i omkretsen av objektet, tettheten i segmenteringen etc. Disse kan igjen brukes for å estimere objektets lengde, bredde og orientering. Se kapittel 9 for mer om dette. Skyggen kan også gi verdifull informasjon om målets geometriske egenskaper hvis den gjenspeiler objektets form.

8.3.2 Radiometriske egenskaper

Disse egenskapene sier noe om RCS i de segmenterte pikslene. Det kan være gjennomsnittsverdi, standardavvik, median, entropi, skjevhet i fordelinger, sterkeste spredere etc.

8.3.3 Linjedeteksjon

Flere transformasjoner kan brukes for å detektere linjer i et bilde. De detekterte linjene kan brukes direkte som egenskaper, eller som ledd i å estimere aspektvinkelen, lengden og bredden til målet. To vanlige transformer er Hough og Radon. Hough virker på et binært bilde, mens Radon kan brukes direkte på et intensitetsbilde. Se kapittel 9.1 for en bedre beskrivelse av Radon-transformen. [16] foreslår også å bruke et tilfeldig Markov-felt for å detektere linjer i SAR-bilder.

8.3.4 Andre egenskaper

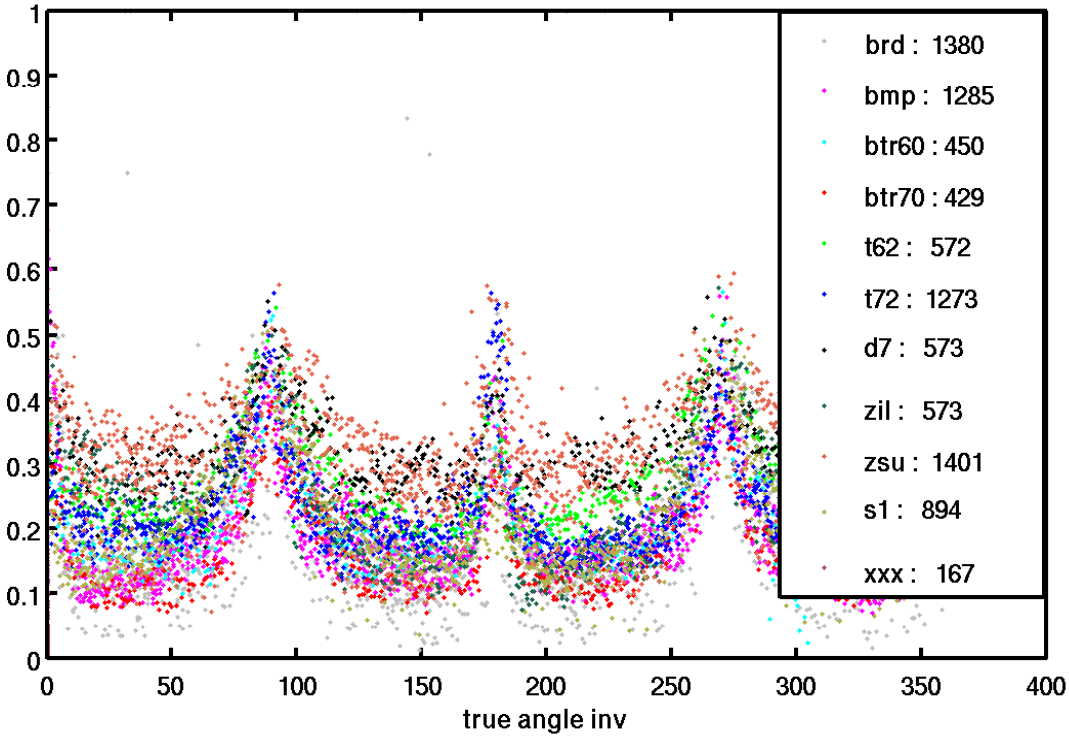
En rekke andre egenskaper er også blitt foreslått i ulike artikler. [17] har brukt Fourier-koeffisienter som inngangsdata i et neuralt nettverk. Wavelet-koeffisienter, polarimetriske egenskaper og fraktaler er andre eksempler. Disse vil bli testet i senere faser av SOBEK.

8.4 Komponent-analyse

For å redusere antall dimensjoner i egenskapsvektoren kan man prøve å lage en lineær kombinasjon av elementene. Principal Component Analysis (PCA) er en transformasjon som søker å finne den lineære kombinasjonen som best reprensenterer dataene på en 'mean-square'-måte. Denne metoden tar imidlertid ikke nødvendigvis hensyn til de egenskapene som best skiller klassene fra hverandre. Multiple Discriminant Analysis (MDA) gjør nett-opp dette. En annen metode, Independent Component Analysis (ICA), finner de retningene i egenskapsrommet som er mest uavhengige av hverandre. Man kan også evt. bruke såkalte selvstendige grupperingsalgoritmer for å redusere dimensjonaliteten på problemet.

8.5 Normalisering av data

De fleste klassifikatorer vil virke bedre hvis dataene normaliseres. Dette er for at ikke noen av egenskapene skal dominere over de andre fordi de måles i andre enheter. Det er vanlig å transformere inngangsdataene til en klassifikator slik at de har 0 i middelverdi og 1 i standardavik. En annen måte å normalisere på som noen ganger brukes i neurale nettverk er å transformere inngangsdataene til å ligge i intervallet $[-1, 1]$.



Figur 9.1: Egenskaper ved de forskjellige klassene kan ofte være avhengige av aspektvinkelen mellom radaren og målet. Her er målets utstrekning i SAR-bildet plottet mot aspektvinkel med fargekoding for de forskjellige klassene i MSTAR-datasettet. Vi ser markante hopp ved 0, 90,180 og 270 grader.

9 Estimering av geometriske egenskaper

Tilbakespredningen fra et mål er sterkt avhengig av aspektvinkelen mellom radaren og målet. Et eksempel på dette kan sees i figur 9.1. Her er tettheten i segmenteringen for de forskjellige klassene i treningssettet plottet mot den sanne aspektvinkelen. Verdiene gjør markante hopp ved 0, 90,180 og 270 grader. Aspektvinkelen er dermed viktig å estimere når man driver målgjenkjenning i SAR-bilder. For å utnytte dette i en klassifikator kan kan f.eks dele opp treningsdataene i mindre vinkelintervaller der verdiene er relativt konstante. Aspektvinkelen kan imidlertid være vanskelig å estimere, spesielt hvis målet er forsøkt skjult. Det at tilbakespredningen er så varierende gjør at segmenteringen ofte heller ikke vil gjenspeile objektets virkelige form. Det kan derfor være vanskelig å finne nøyaktiske estimater av geometriske egenskaper ved objektet som lengde og bredde.

I en kontrollert datainnsamling kan man logge målets aspektvinkel i hvert bilde, for senere å se hvilken effekt denne har på klassifikasjonen. Dette gir mulighet til å estimere aspektvinkelen bare basert på bildene, og for deretter å analysere hvor nært man kom det riktige svaret. I MSTAR-datasettet er aspektvinkelen logget, og dette datasettet er dermed fint å bruke for å lage en vinkelestimator. I en virkelig situasjon må man basere seg på estimerte verdier for aspektvinkelen, og det er derfor viktig at dette estimatet er så godt som mulig. Man kan også teste ut forkjellige klassifikatorer og se hvor stor innvirkning de forskjellige aspektvinkelestimatene har på resultatet. De følgende delkapitlene vil se på forskjellige estimeringsmetoder.

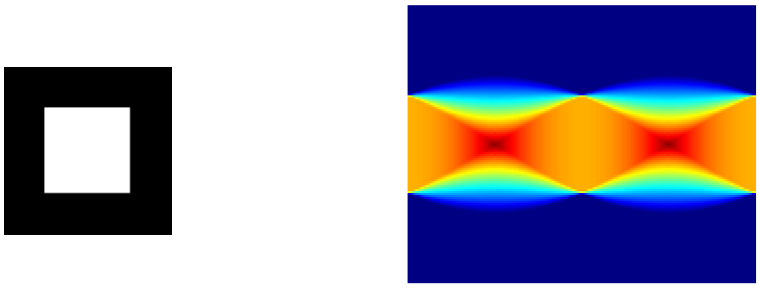
9.1 Radon-transformen

Radon-transformen projiserer intensiteten i et bilde langs linjer i forskjellige orienteringer. Dette gir et mål på hvor mye energi som samler seg i ulike vinkler og dermed den mest sannsynlige aspektvinkelen. Den generelle formelen for Radon-transformen er

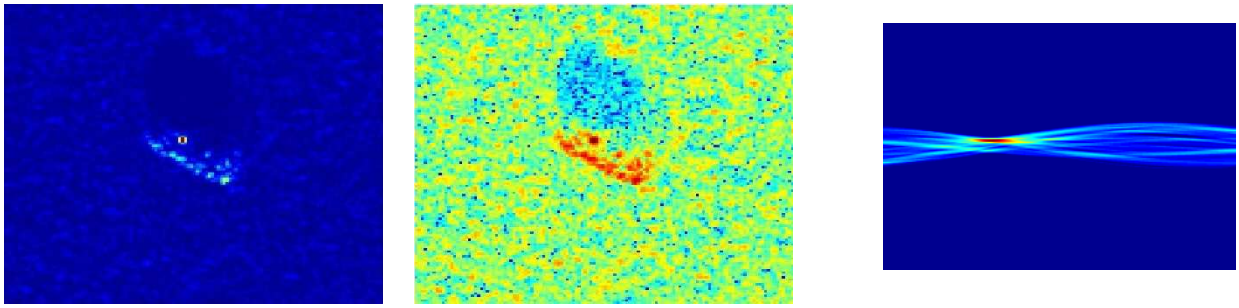
$$R_\theta(x') = \int_{-\infty}^{\infty} f(x'\cos\theta - y'\sin\theta, x'\sin\theta + y'\cos\theta)dy' \tag{9.1}$$

der

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{9.2}$$



Figur 9.2: Et kvadrat og Radon-transformen av kvadratet

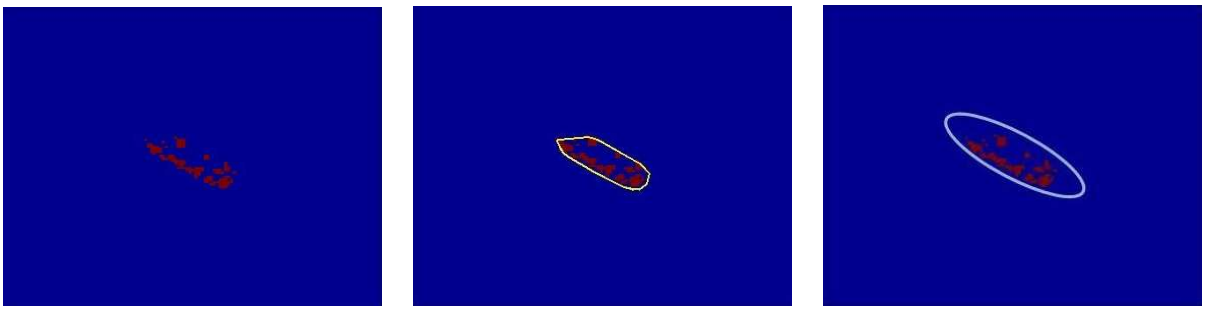


Figur 9.3: Amplitude-bilde, logaritmisk bilde og Radon-transformen av det segmenterte bildet

I figur 9.1 ser vi Radon-transformen av et kvadrat. De sterkeste punktene og de smaleste båndene dukker opp i vinkelene 45 og 135 grader. Både det sterkeste punktet og det smaleste båndet kan brukes for å estimere aspektvinkelen. I tillegg til å finne aspektvinkelen kan Radon-transformen brukes til å detektere linjer i et bilde. Disse kan igjen brukes som egenskaper i en klassifikator. I virkelige data har vi testet Radon-transformen både med segmenterte bilder og originale logaritmiske bilder. Figur 9.3 viser amplitude-bildet, det logaritmiske bildet samt Radon-transform av det segmenterte bildet.

9.2 Andre estimeringsmetoder

Den ellipsen som best omslutter de segmenterte pikslene danner en vinkel med x-aksen i bildet. Denne kan brukes som et estimat av aspektvinkelen. Den konvekse omslutningen til de segmenterte pikslene kan også finnes. Et annet estimat av aspektvinkelen kan dermed fåes ved å se på orienteringen til denne istedenfor orienteringen til ellipsen. Den konvekse omslutningen kan videre Radon-transformeres for å finne nok et vinkelestimat. Figur 9.4 viser de segmenterte pikslene, den konvekse omslutningen samt den ellipsen som best omslutter de segmenterte pikslene.



Figur 9.4: De segmenterte pikslene, den konvekse omslutningen, samt den ellipsen som best omslutter de segmenterte pikslene

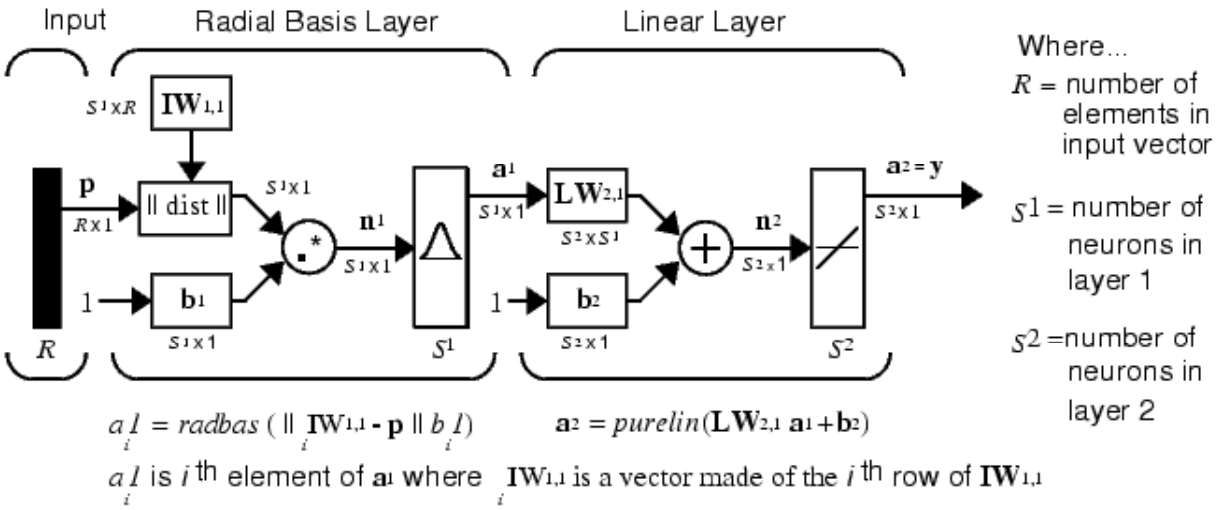
9.3 Neurale nett for vinkelestimering

De to metodene for vinkelestimering gir flere ulike estimater. Problemet blir å bestemme hvilket man skal stole på. Ingen av metodene har vist seg å gi det beste estimatet i alle tilfeller. En mulighet for å finne et mest mulig robust estimat kan være å fusjonere estimater fra forskjellige metoder. Dette kan bl.a. gjøres ved hjelp av et neuralt nett. Når man fusjonerer estimater må man finne de riktige vektene for hvert estimat. Et neuralt nett er i stand til å finne de optimale vektene for en slik fusjonering. Vektene vil imidlertid være tilpasset treningssettet, og trenger ikke nødvendigvis å virke godt på nye data. Dette har vi forsøkt å implementere.

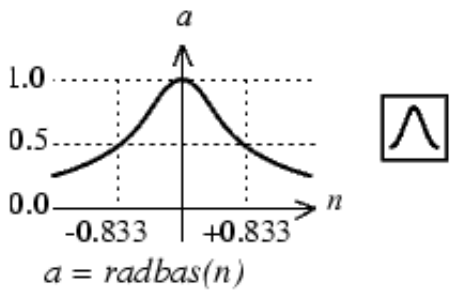
Nettet som ble valgt kalles Generalized Regression Neural Network (GRNN). Dette er en versjon av Radial Basis Networks. Disse nettene brukes ofte til funksjonsapproksimering (regresjon) istedenfor klassifikasjon. GRNN har et 'radielt basis'-lag og et lineært lag. Det lineære laget gjør det mulig å få vilkårlige utgangsverdier. Det skjulte laget inneholder mange neuroner, like mange som inngangssampler. GRNN kan approksimere en hvilken som helst kontinuerlig funksjon gitt at den har nok neuroner i det skjulte laget. Oppsettet vises i figur 9.5.

Figur 9.6 viser transferfunksjonen til et neuron. Her er $radbas(n) = e^{-n^2}$. n er distansen mellom inngangs- og vektvektoren multiplisert med $0.833/s$ der s er spredning og en parameter brukeren kan sette. Dette gjør at a krysser 0.5 ved $\pm s$. s må være så stor at neuroner responderer kraftig i overlappende områder av inngangs-rommet. Hvis s er for liten blir funksjonen støyete, og hvis den er for stor får vi for mye interpolasjon. I praksis tester man gjerne en del verdier for spredningen og velger den som gir best resultat.

I tillegg til de ulike aspektvinkelestimatene ble egenskaper ved objektene som antas å ha betydning for troverdigheten til de ulike estimatene tatt med. De parametrene som er valgt i tillegg til vinkelestimatene er elevasjonsvinkel, eksentrisitet, utstrekning, soliditet, antall



Figur 9.5: Generalized Regression Neural Network, figuren er hentet fra Matlab-dokumentasjonen



Figur 9.6: Radial Basis Neuron, figuren er hentet fra Matlab-dokumentasjonen

piksler i segmenteringen og antall piksler i den konvekse omslutningen. Eksentrisitet vil si hvor eksentrisk den ellipsen som best omslutter alle de segmenterte pikslene er. Verdien kan ligge mellom 0 og 1. Hvis eksentrisiteten er 0 har vi en sirkel, mens hvis den er 1 er det bare en linje. Teorien er at hvis eksentrisiteten er stor, kan man stole mer på vinkelestimatet man får gjennom ellipsen enn om eksentrisiteten er liten. Hvis man legger et rektangel rundt de segmenterte pikslene og ser hvor tett pikslene faktisk fyller dette rektangelet, får man utstrekning. Det er med andre ord et mål på hvor 'firkantet' området er og hvor tettpakket dette isåfall er. Soliditet vil si at man gjør det samme med den konvekse omslutningen. Er disse verdiene store kan man kanskje stole mer på estimatene man får fra Radon-transformen enn om de var små. Så blir det opp til nettet å finne sammenhenger mellom disse parametrene og estimatene som ikke er opplagte for et menneske. Alle inngangsverdiene i nettet blir normalisert.

Resultatene viser at vinkelestimatet blir bedre med det neurale nettet enn ved å bare bruke de individuelle estimatene. Figur 9.7 viser resultatene ved å kjøre dette bare på treningssettet. Nederst til venstre er resultatet fra nettet, mens de andre plottene er de ovennevnte vinkelestimatene. Fordi vi her bare ser på treningsdataene er det som forventet at RMS-verdien blir lav.

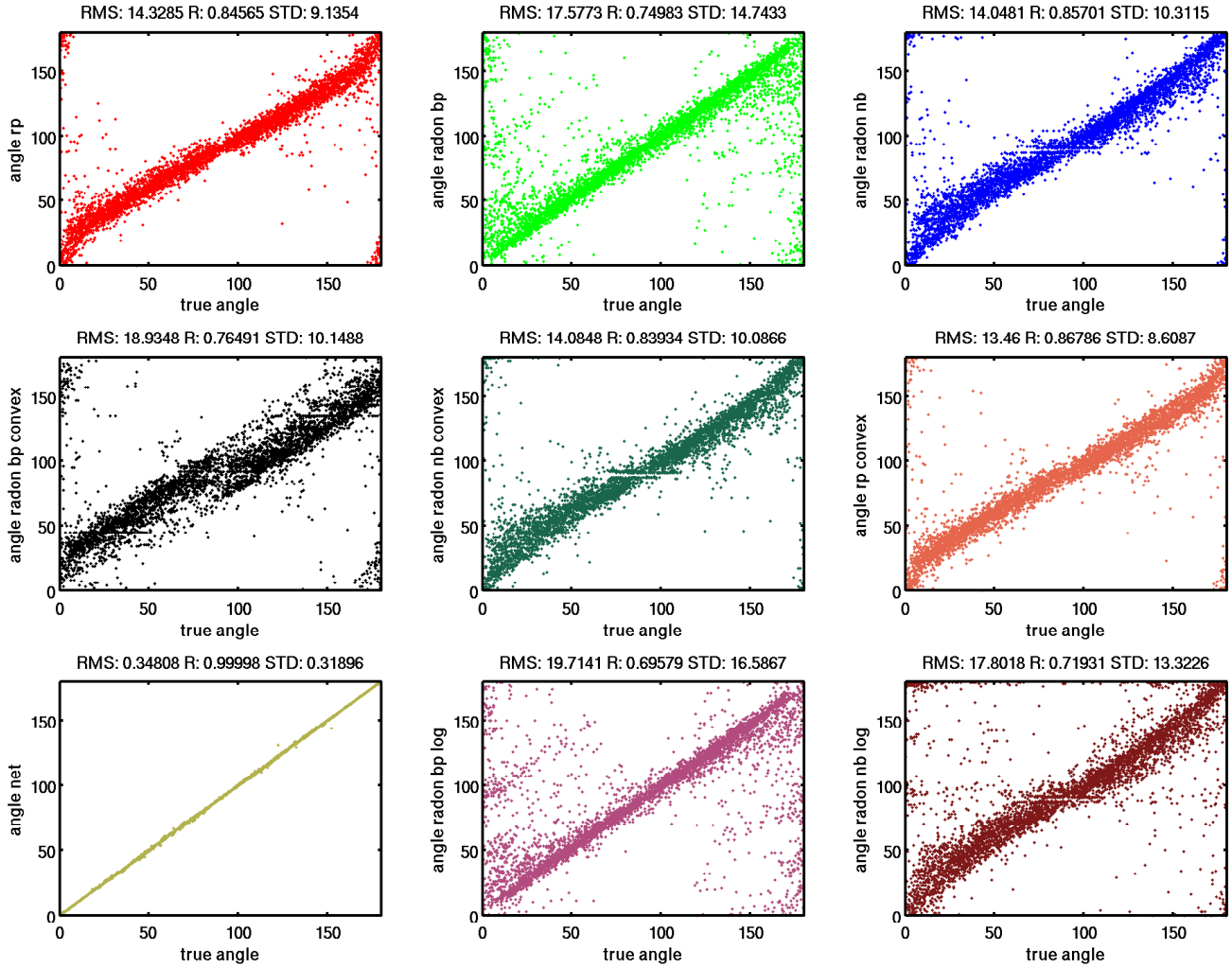
Spørsmålet er om dette nettet klarer å generalisere, dvs. å estimere aspektvinkelen godt for nye data. Resultatet for testsettet er som forventet ikke like bra som for treningssettet, men likvel bedre enn for de individuelle estimatene. Figur 9.8 viser resultatet for testsettet.

Det er stor variasjon i reultatene klassene i mellom i MSTAR-datasettet. Nettet har vanskeligst for å estimere aspektvinkelen til BRD, mens lettest er det for T62.

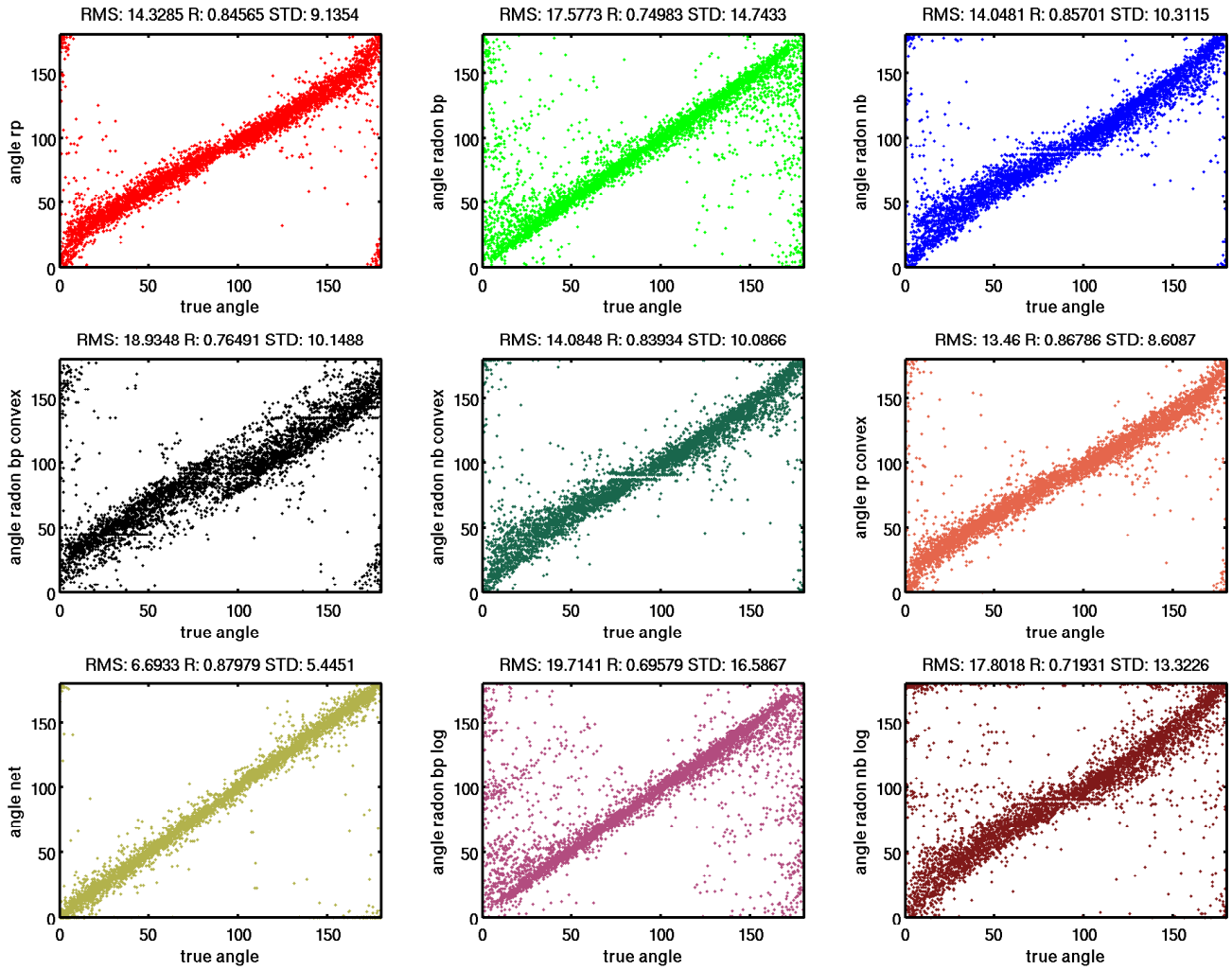
Siden dette samlede estimatet for aspektvinkel er bedre enn noen av de individuelle, brukes det videre i klassifikatorene. Det er viktig å merke seg at kvaliteten på aspektvinkelestimatet vil avhenge av kvaliteten på segmenteringen.

9.4 Lengdeestimering med neuralt nettverk

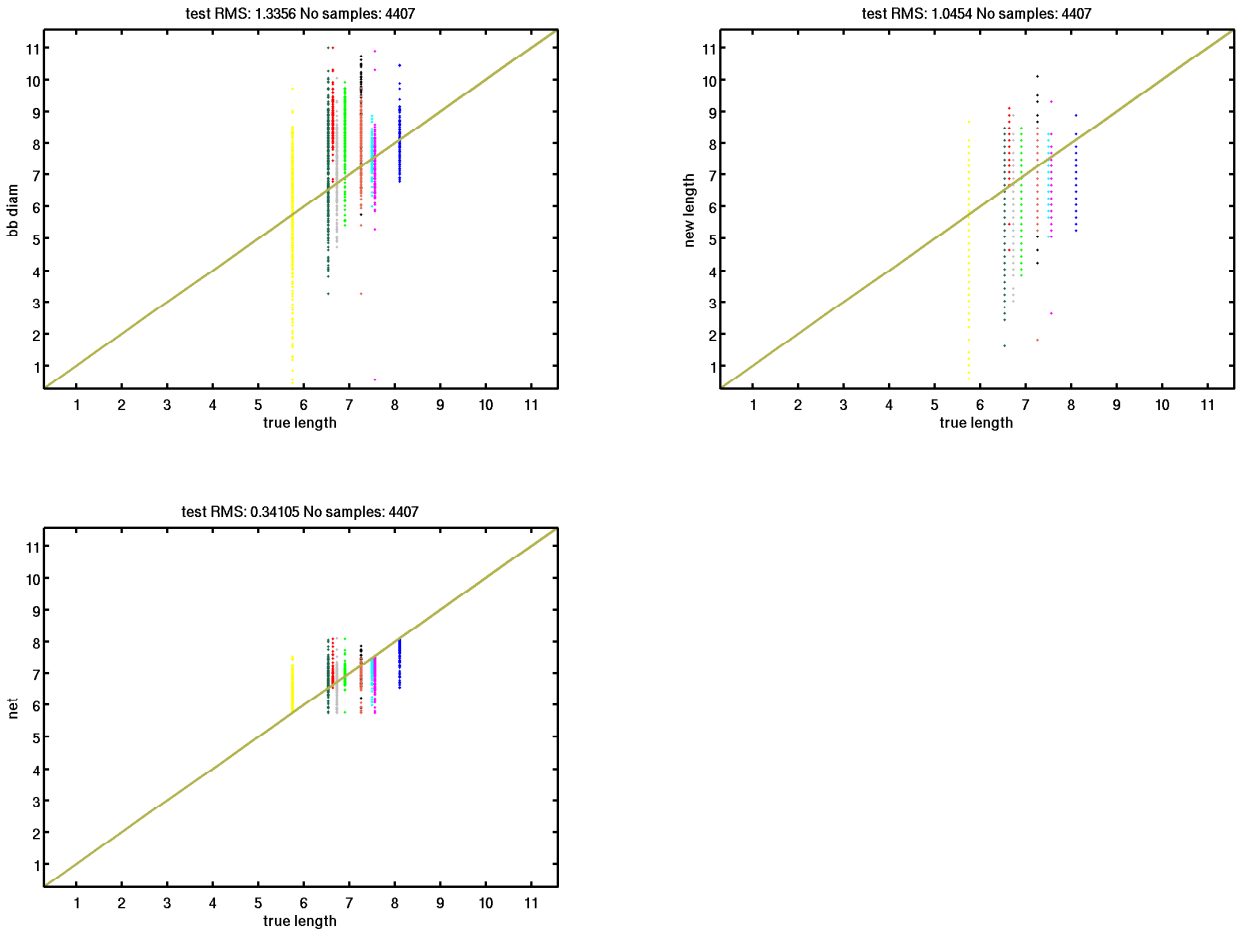
For å estimere objektets lengde ble samme nettverk som for vinkelestimering brukt, men med andre inngangsverdier. Først finner vi to forskjellige lengeestimater. Et estimat er den lengste lengdeaksen i ellipsen som best omslutter de segmenterte pikslene. Et annet fåes ved å rotere bildet med en vinkel lik den som ble estimert med vinkelestimeringsnettet. Deretter summeres antall rader. De parametrene ved objektene som vi antar har betydning for troverdigheten til de ulike lengdeestimatene er elevasjonsvinkel, eksentrisitet, utstrekning, soliditet, antall piksler i segmenteringen, antall piksler i den konvekse omslutningen, dynamisk område for pikselverdiene, størrelsen og diameteren på det rektangelet som akkurat



Figur 9.7: Forskjellige metoder for å estimere aspektvinkelen i treningsdataene er plottet mot sann vinkel. Estimatet fra nettet vises nederst til venstre. Hvert plott inneholder også RMS-verdi, regresjonsverdi og standardavvik.

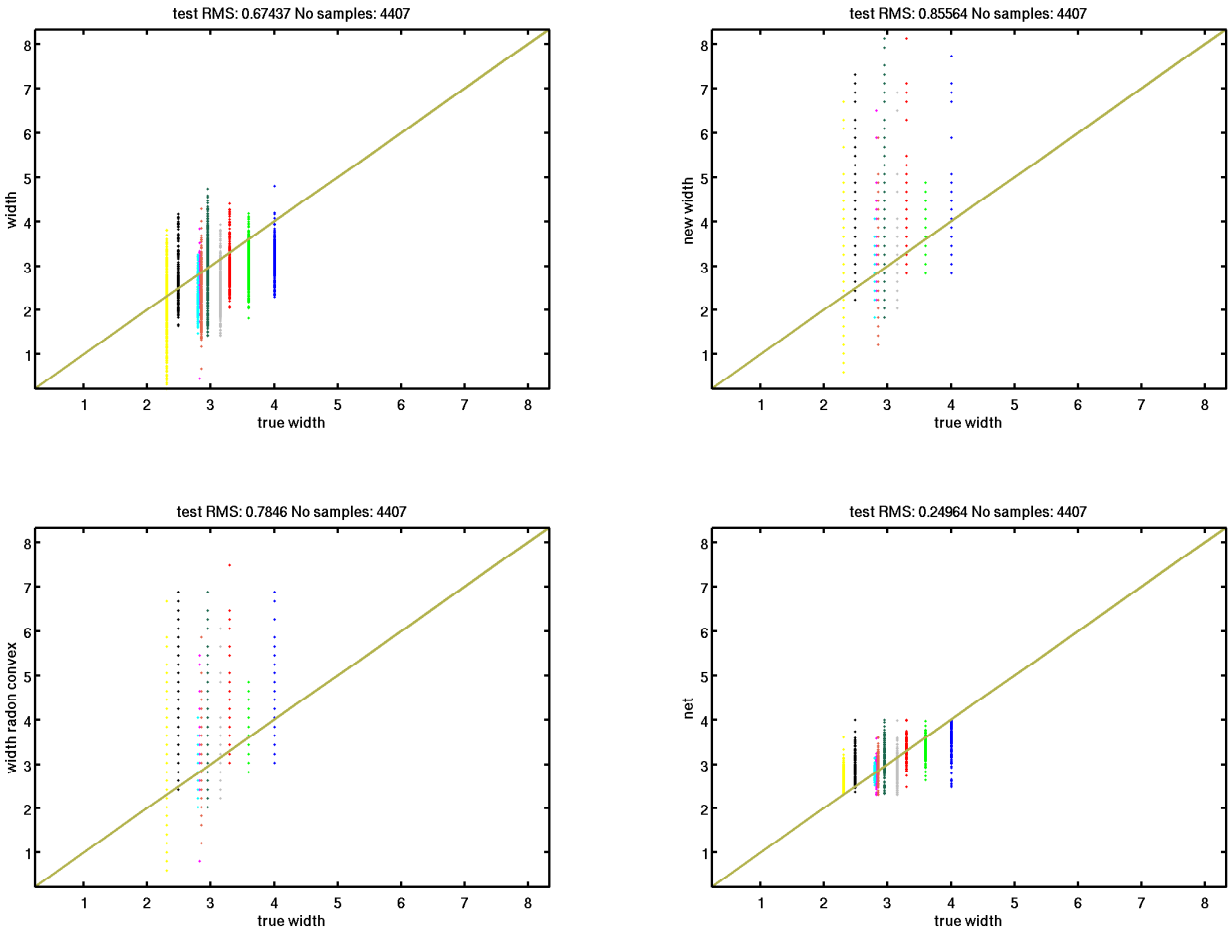


Figur 9.8: Forskjellige metoder for å estimere aspektvinkelen i testdataene er plottet mot sann vinkel. Estimatet fra nettet vises nederst til venstre. Hvert plott inneholder også RMS-verdi, regresjonsverdi og standardavvik.



Figur 9.9: Forskjellige metoder for å estimere lengden i testdataene er plottet mot sann lengde. Punktene er fargekodet etter klasse i MSTAR-datasettet. Estimatet fra nettet vises nederst.

dekker pikslene, samt den aspektvinkelen som ble estimert gjennom vinkelestimeringsnettet. Resultatet vises i figur 9.9. Her ser vi at RMS-verdien blir lavere ved å bruke et nett, men de ulike lengdeestimatene er i seg selv så unøyaktige at det er vanskelig å skille mellom klassene uansett. Det må derfor jobbes mer med å få disse estimatene til å bli bedre i utgangspunktet. Det kan bl.a. gjøres ved å forbedre segmenteringen, preprosesseringen og selve estimeringsmetodene.



Figur 9.10: Forskjellige metoder for å estimere bredden i testdataene er plottet mot sann bredde. Punktene er fargekodet etter klasse i MSTAR-datasettet. Estimatet fra nettet vises nederst til høyre.

9.5 Breddeestimering med neuralt nettverk

For å estimere objektets bredde har vi igjen brukt samme nettverk som for vinkelestimering, men med andre egenskaper som innganngsverdier. Først er tre forskjellige breddeestimater funnet. Ved å måle utstrekningen på det smaleste båndet i Radon-transformen til den konvekse omslutningen får man et estimat av objektets bredde. Videre brukes den korteste lengdeaksen i ellipsen som best omslutter de segmenterte pikslene som et annet estimat. Et siste fåes ved å rotere bildet med en vinkel lik den som ble estimert med vinkelestimeringsnettet. Deretter summeres antall kolonner. Her antar vi at de samme parametrene for å vurdere troverdigheten til estimatene som ble brukt for lengdeestimering kan brukes. I figur 9.10 vises resultatene.

Her ser vi at også RMS-verdien blir lavere ved å bruke et nett, men på samme måte som for lengdeestimatene er de ulike breddestimatene i seg selv så unøyaktige at det er vanskelig å skille mellom klassene uansett.

10 Klassifikatorer

Mange ulike klassifikatorer har blitt utviklet gjennom tidene. I dette kapittelet beskrives de algoritmene som er implementert så langt i prosjekt SOBEK. Samtidig nevnes noen flere som vi vil forsøke å implementere etterhvert. Først vil Bayes generelle beslutningsregel bli beskrevet. Denne danner grunnlaget for flere klassifikatorer.

10.1 Bayes regel

Hvis man kjenner de underliggende sannsynlighetsfordelingene til de forskjellige klassene i et klassifikasjonsproblem kan det vises at den beste klassifikatoren vil følge Bayes regel (se [15, kapittel 2]). Denne bruker følgende uttrykk for a posteriori sannsynlighet for at riktig klasse er w_j når x er blitt observert:

$$P(w_j \mid x) = \frac{p(x \mid w_j)P(w_j)}{p(x)} \tag{10.1}$$

der

$$p(x) = \sum_{j=1}^{c} p(x \mid w_j)P(w_j) \tag{10.2}$$

Her er $P(w_j)$ a priori sannynligheten for at vi har klasse w_j og $p(x \mid w_j)$ den betingede sannsynligheten for å observere x når den virkelige klassen er w_j. x er den tilfeldige egenskapsvektoren vi måler og $p(x)$ er den samlede sannsynligheten for å observere x, uavhengig av klasse. c er antall klasser. For å få en klassifikator ut av dette trenger vi en beslutningsregel. Denne sier i hvilke tilfeller man bør gjette på klasse w_j for å minimere sannsynligheten for å feilklassifisere. I et toklasseproblem vil beslutningsregelen ved å følge Bayes teori dermed bli: Velg w_1 hvis $P(w_1 \mid x) > P(w_2 \mid x)$, ellers velg w_2.

10.2 Parametriske eller ikke-parametriske metoder

Det er to hovedmåter å tilnærme seg et klassifikasjonsproblem på: parametriske og ikke-parametriske metoder. I de parametriske metodene antar man eller kjenner formen på de

underliggende fordelingene. I de aller fleste virkelige problemer vet vi ikke noe om a priori sannsynligheter eller om sannsynligheten for å observere x. Disse fordelingene må dermed estimeres. Hvis man antar form på fordelingene, er det bare fordelingenes parametre som må estimeres. Disse kan estimeres på ulike måter. Problemet med parametriske metodene er at hvis vi antar feil underliggende fordelinger vil klassifikatoren ha dårlig ytelse. I de ikke-parametriske metodene bryr man seg ikke om de underliggende fordelingene, men begynner å lage beslutningsregler med en gang basert på den observerte x.

10.3 Minimum feilrate

Minimum feilrate-klassifikasjon er en videreføring av Bayes regel der man innlemmer en feilfunksjon. Den er dermed et eksempel på en parametrisk metode. Man kan vekte hvor kritisk det er å feilklassifisere til bestemte klasser. Hvis alle feilklassifikasjoner er like alvorlige kan man bruke den såkalte 'symmetriske' eller 'null-en' feilfunksjonen. Den er gitt ved

$$\lambda(\alpha_i \mid w_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \quad i, j = 1, ..., c. \tag{10.3}$$

Her er α de forskjellige beslutningene, og $\lambda(\alpha_i \mid w_j)$ er tapet man får ved å bruke beslutning α_i når den sanne klassen er w_j. Denne betingelsen gir følgende beslutningsregel for å minimere den totale risikoen (se [15, kapittel 2.3]): Velg w_i hvis $P(w_i \mid x) > P(w_j \mid x)$ for alle $j \neq i$. c er antall klasser. Med denne feilfunksjonen vektes alle feilklassifikasjoner likt. Andre feilfunksjoner vil gi andre beslutningsregler.

Ved å anta at de underliggende fordelingene er normale kan man forenkle diskriminantfunksjonene. Videre kan man forenkle ennå mer ved å anta at alle klassene har samme kovariansmatrise. Her må isåfall bare forventning og varians estimeres fra treningsdataene.

Fordelingens parametervektor $\boldsymbol{\theta}$ kan igjen estimeres på forskjellige måter, som f.eks Maximum-likelihood (ML) eller Bayesisk estimering. ML går ut på å finne den $\boldsymbol{\theta}$ som maksimerer sannsynligheten for å observere de egenskapene man faktisk har obervert. Bayesisk estimering ser på $\boldsymbol{\theta}$ som tilfeldige variable med en gitt forhåndsfordeling. Etter å ha observert noen egenskaper kan denne konverteres til en a posteriori fordeling. Ved å observere flere og flere sampler vil denne fordelingen vanligvis bli skarpere og skarpere og ha sitt toppunkt nær den virkelige verdien av $\boldsymbol{\theta}$. Mer presist sagt vil estimatet bli en vektet kombinasjon av vår opprinnelige antatte fordeling for $\boldsymbol{\theta}$ samt de observasjonene man gjør. ML kan sees på som Bayesisk estimering der forhåndsfordelingen er uniform, dvs. vi har ingen antakelser

om verdien på $\boldsymbol{\theta}$ før vi observerer samplene. Bayesisk estimering er derfor mer regnekrevende, men kan gi bedre estimater enn ML fordi man bruker mer informasjon enn bare dataene.

10.4 k-nærmeste nabo

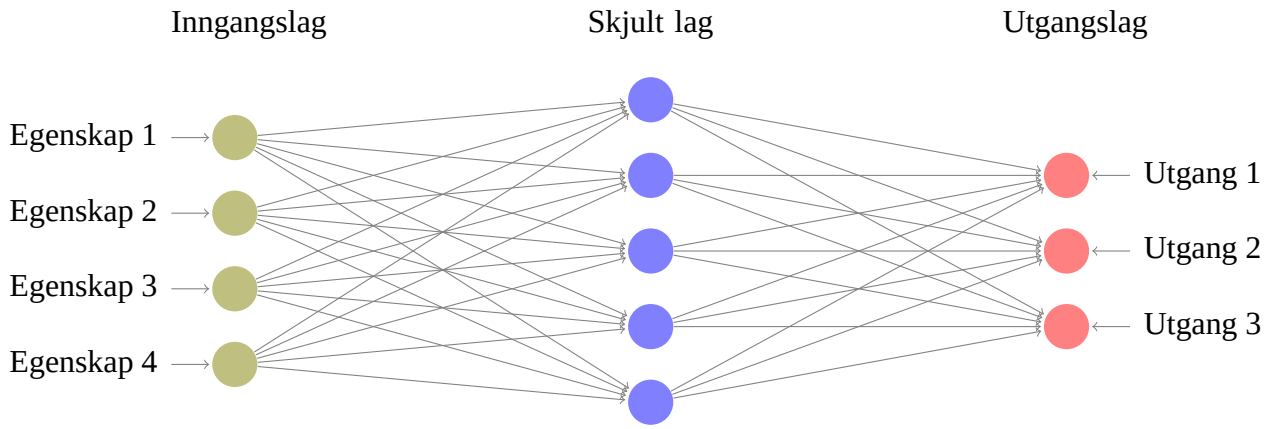
k-nærmeste nabo-metoden prøver ikke å estimere de underliggende fordelingene, men går direkte til å finne diskriminantfunskjoner. Den er en underklasse av Parzen-vinduer (se [15, kapittel 4]), der vindu-funksjonen velges så stor at den til enhvert tid inneholder k sampler. Man velger altså de k nærmeste naboene til testsamplet med et visst avstandsmål. Testsampelet blir klassifisert til den klassen de fleste av de k treningssamlplene hører til. Et vanlig avstandsmål er Euklids distanse, men enhver norm som tilfredsstiller en del krav kan brukes. k velges odde for at det ikke skal bli to klasser som får like mange sampler. k-nærmeste nabo-metoden har en stor ulempe, og det er at den er regnekrevende. For hvert nye sample må man søke gjennom alle treningssamplene for å finne de k nærmeste. Det er imidlertid mulig å komme rundt dette problemet ved å bruke parallelliserings- og/eller søketre-teknikker når algoritmen skal implementeres. Dette er ennå ikke forsøkt i prosjekt SOBEK.

Fordi tilbakespredning i SAR-bilder er sterkt avhengig av aspektvinkel kan algoritmens ytelse forbedres ved å bare undersøke de nærmeste naboene i et lite vinkelspenn rundt aspektvinkelen til testobjektet. Før man kan gjøre dette må man imidlertid forsikre seg om at man har samplet rommet med aspektvinkler godt nok, slik at alle spennene har representasjoner fra hver klasse i treningssettet. Selv om det er noen representasjoner fra hver klasse i hvert vinkelspenn bør det være en viss mengde for at ytelsen til klassifikatoren skal bli god nok. I tillegg til å kunne forbedre ytelsen forkortes også prosesseringstiden ved å begrense søket til visse vinkelintervaller.

En annen måte å bruke vinkelinformasjonen på er å dele inn i faste vinkelintervaller som gjenspeiler egenskapenes vinkelavhengighet. Her vil man kunne bruke større intervaller enn ved å lete i et lite område rundt testsampelets egen estimerte vinkel.

10.5 Neurale nett

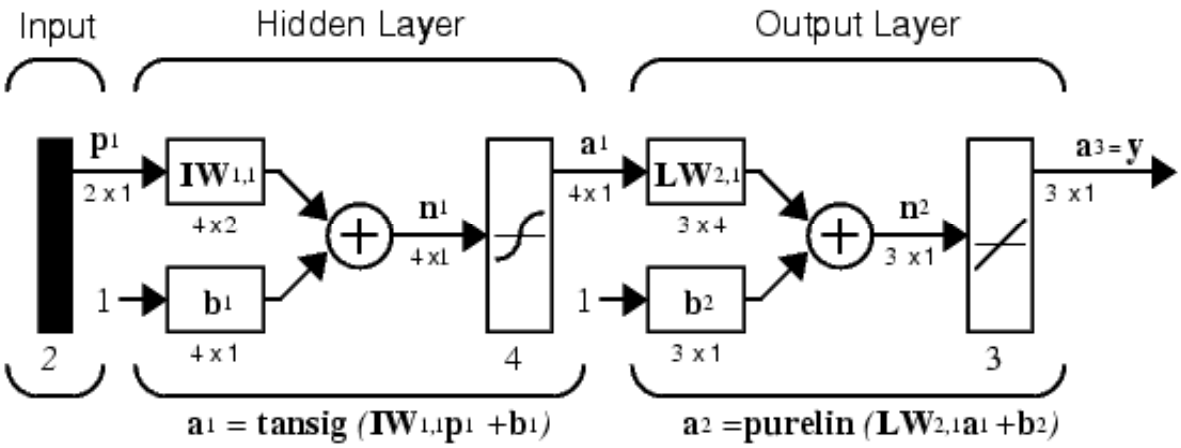
Neurale nettverk er et annet eksempel på en ikke-parametrisk klassifikasjonsmetode. Disse nettverkene er inspirert av biologiske neuroner. Nettverket trenes opp til å kjenne igjen visse mønstre. Det er utviklet mange forskjellige neurale nettverk for å løse en rekke ulike problemer, bl.a. klassifikasjon, regresjon, prediksjon m.m. I figur 10.1 ser vi et typisk



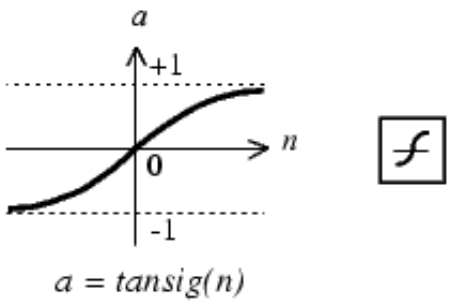
Figur 10.1: Eksempel på et generelt neuralt nett. Her består inngangsvektoren av fire forskjellige egenskaper og det er tre mulig utgangsklasser. Det er et skjult lag med fem noder. Antall forbindelser (vekter) kan sees på som problemets frihetsgrader.

neuralt nett der man sender inn fire egenskaper per sample. I dette problemet er det tre klasser, og dermed tre utgangsnoder. Det kan være et eller flere skjulte lag. Jo flere skjulte lag, jo mer ulineære problemer kan nettet løse. Det er vektene mellom alle nodene som gjør at nettverket fungerer. For å løse et klassifikasjonsproblem har vi valgt å bruke et såkalt tilbakepropageringsnettverk. Dette vises i figur 10.2. Tilbakepropagasjon er en rekursiv metode der feilen man får ved å sende treningssamplene gjennom nettverket tilbakepropageres for å finne bedre vekter i neste steg. For å finne de forbedrede vektene brukes gradient-søkmetoder. Også dette kan gjøres på flere forskjellige måter. Man kan imidlertid ikke være sikker på at algoritmen ikke blir fanget i et lokalt minimum, og det kan derfor lønne seg å teste algoritmen med forskjellige startverdier. Spesielt med flere skjulte lag er det en stor fare for dette. Hvor stor steglengden skal være i hver runde kan også velges og vil ha noe å si for konvergens av algoritmen. Fordi man sjelden vet noe om de underliggende fordelingene kan en slik ikke-parametrisk metode være et godt valg. En ulempe med neurale nett er at de kan tilpasse seg et treningssett for godt, og dermed ikke generalisere så bra på nye data. Dette kan man forsøke å unngå ved å regne ut ytelsen på et valideringssett under treningen. Treningen stoppes hvis forskjellen mellom nettets ytelse på treningssettet og valideringssettet overstiger en viss grense. I tillegg finnes det teknikker for å fjerne de minst viktige vektene i nettverket.

Antall noder i de forskjellige lagene spiller en rolle for resultatet. Det finnes ingen fasit på hvor mange noder man bør ha med. Det er vanlig å teste med ulike antall, men det finnes også en tommerfingerregel som sier at man bør velge antall noder slik at antall vekter i nettverket ikke overstiger $\frac{n}{10}$ der n er antall treningssampler.



Figur 10.2: Tilbakepropageringsnettverk, figuren er hentet fra Matlab-dokumentasjonen



Figur 10.3: Tan-sigmoid transferfunksjon, figuren er hentet fra Matlab-dokumentasjonen

10.6 Andre klassifikatorer

Det finnes en mengde andre klassifikatorer vi ennå ikke har forsøkt å implementere. Dette er f.eks Support Vector Machines, skjulte Markov-modeller, Random Forests, klassifikasjonstrær, grammatiske metoder, stokastiske metoder eller kombinasjoner av disse (datafusjon). Datafusjon er et helt eget felt der det finnes mange måter å bestemme vektene mellom resultatene av de ulike klassifikasjonene. I tillegg kan Minimum feilrate-metoden implementeres med andre antatte underliggende fordelinger. Det finnes også flere måter å implementere tilbakepropageringsnett på.

11 Evaluering

Evaluering er en viktig del av ATR-prosessen. For å kunne gjenskape en god klassifikasjon er det viktig at parametre som ble brukt tas vare på sammen med et mål på kvaliteten. Prosjekt SOBEK bruker en database der alle nødvendige opplysninger om hver test blir lagret.

11.1 Forvirringsmatriser og ROC

For å gi et mål på hvor godt en klassifikator virker, er det vanlig å sette opp en forvirringsmatrise. Et eksmpel på en slik matrise finnes i tabell 11.1. Her er det tatt utgangspunkt i MSTAR-datasettet, men verdiene er fiktive. Objekter av klassene i hver rad har blitt klassifisert som den kolonnen de havner i. Tallene i rødt viser antall korrekte klassifikasjoner, mens de sorte er feilklassifikasjoner. Ideelt sett skulle alle de sorte tallene vært 0. I tillegg til de gyldige klassene (klassene treningssettet inneholder) må vi ha med en forkastelsesklasse (ukjent). Der havner mål som ikke passer inn i noen av de andre klassene. Man setter opp en terskelverdi for hvor langt unna de gyldige klassene et mål må være før det blir klassifisert som ukjent. Denne verdien kan varieres, og man får forskjellige resultater ved å bruke ulike terskelverdier. Det vil si at for hver terskelverdi man tester får man en ny forvirringsmatrise. I tillegg til testdata av samme klasser som i treningsdataene, bør klassifikatoren også testes på objekter av en ukjent klasse. Dette gir nederste raden i forvirringsmatrisen. Når disse ukjente objektene blir klassifisert som en av klassene i treningssettet kalles det falsk alarm. Helst skulle alle de ukjente objektene havnet i ukjent-klassen. Ofte vil forvirringsmatriser som viser mye korrekt klassifikasjon også ha mange falske alarmer. Målet for en klassifikator er å optimalisere begge disse kriteriene, dvs. gi flest mulig korrekte klassifikasjoner og samtidig få falske alarmer. Det er dermed ikke nok å rapportere en forvirringsmatrise for å

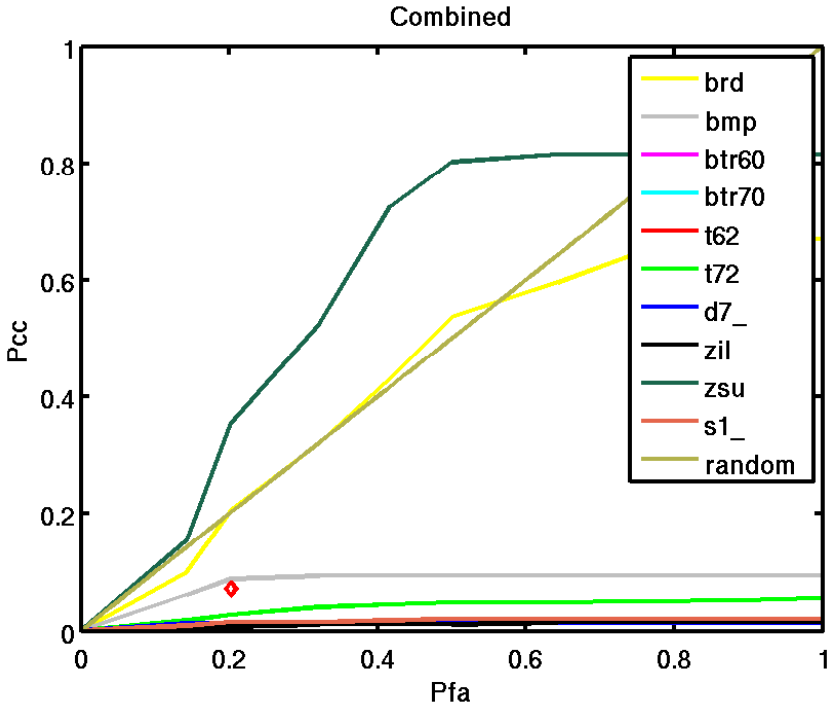
vise kvaliteten på en klassifikator, man må også si noe om hvordan resultatene foranderer seg med varierende terskelverdier. Man setter defor opp grafer som viser forholdet mellom korrekt klassifikasjon og antall falske alarmer. Disse grafene kalles ROC-kurver. Det står for 'Receiver Operating Characteristic'. Det er forskjellige måter å fremstille ROC-kurvene på.

I en fremstillingsmåte vises avveiningen mellom å korrekt klassifisere objekter fra en klasse C_k som C_k mot å misklassifisere objekter fra en ukjent klasse som C_k (dvs. falsk alarm). Her er P_{fa} sannsynligheten for falsk alarm og P_{cc} sannsynligheten for korrekt klassifikasjon (tallene i rødt i forvirringsmatrisene). Hver verdi av P_{fa} svarer til en forvirringsmatrise. Hver klasse i treningssettet får en kurve med sin egen farge. Man kan også slå sammen alle klassene og få bare en kurve, men må da se på hver enkelt forvirringsmatrise for å se utviklingen for hver klasse. Figur 11.1 viser et eksempel på en slik graf. Målet er å få P_{fa} så lav som mulig og P_{cc} så høy som mulig. I dette eksempelet er derfor ikke resultatet særlig bra. I tillegg ser vi at de ulike klassene får veldig forskjellige resultater. Den røde prikken på figuren viser det punktet der gjennomsnittet av $P_{cc} - P_{fa}$ for alle klassene er høyest. Linja fra $[0, 0]$ til $[1, 1]$ er det samme som tilfeldig klassifikasjon, og siden noen av kurvene her ligger under denne linja, er dette et eksempel på en svært dårlig klassifikator.

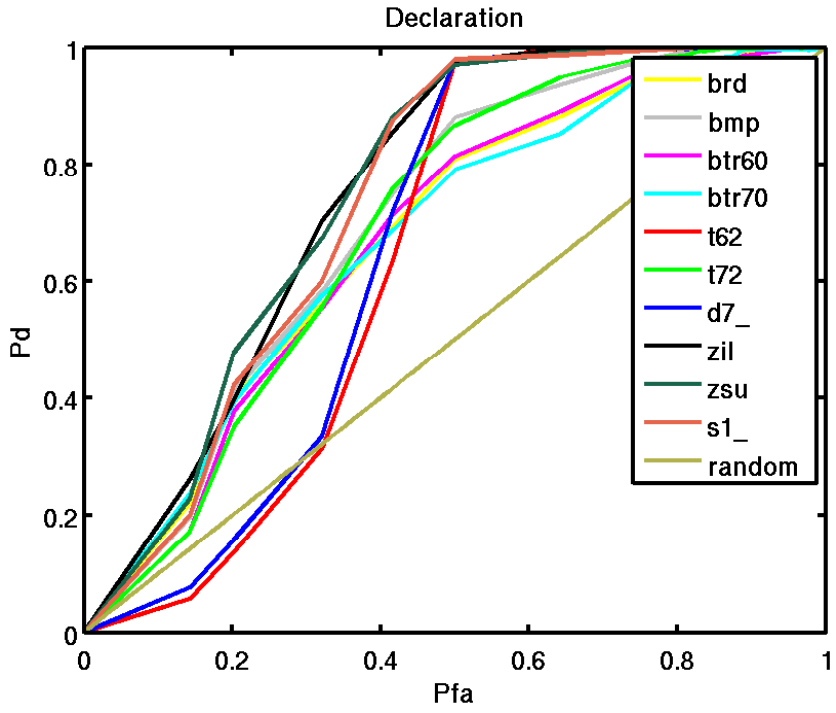
I en annen fremstillingsmåte dekomponerer man det ovennevnte plottet i to andre plott. I det ene plottet vises P_{fa} mot P_d. P_d er sannsynligheten for deklarasjon, dvs. at målet ikke blir klassifisert som ukjent, men som en av klassene i treningssettet. Figur 11.2 viser et eksempel på en slik graf. I det andre plottet vises P_{cc} mot P_d, dvs. sannsynligheten for at de objektene som allerede er blitt deklarert også blir klassifisert med korrekt klasse. Figur 11.3 viser et eksempel på en slik graf. Ved å bruke to plott kommer det litt bedre frem hva som er årsaken til resultatene, om det er deklarasjonen eller klassifikasjonen som har problemer. Det kan være nyttig å se på alle de tre forskjellige plottene (P_{fa}- P_{cc}, P_{fa}-P_d og P_{cc}-P_d) for å virkelig se hva som skjer, og alle tre kommer til å bli brukt videre i rapporten når klassifikatorer skal evalueres.

I operativ bruk velger man ofte en terskelverdi basert på en konstant falsk alarm rate. En operatør vil ofte kanskje ønske å ha en P_{fa} på høyst 0.1. Det vil si at ikke mer enn en tiendel av ukjente objekter skal bli klassifisert som en kjent klasse. Ved å velge en P_{fa} man vil jobbe under kan man presentere et samlet kvalitetsmål på klassifikatoren for den spesielle terskelverdien, nemlig P_{cc}.

Man bør også teste ut klassifikatoren på mål av samme klasse som de i treningsdataene, men fra et helt annet datasett. Ofte vil resultatet da blir dårligere enn om dataene kom fra det samme settet som klassifikatoren ble trent på. Derfor kommer målet som heter '2S1 (uavh)' inn. Dette betyr at målet er uavhengig av treningssettet, og helst bør man ha uavhengige



Figur 11.1: Eksempel på plott av korrekt klassifikasjon mot falsk alarm for klassene i MSTAR-datasettet. Den røde prikken viser punktet der gjennomsnittet av $P_{cc} - P_{fa}$ for alle klassene er høyest.



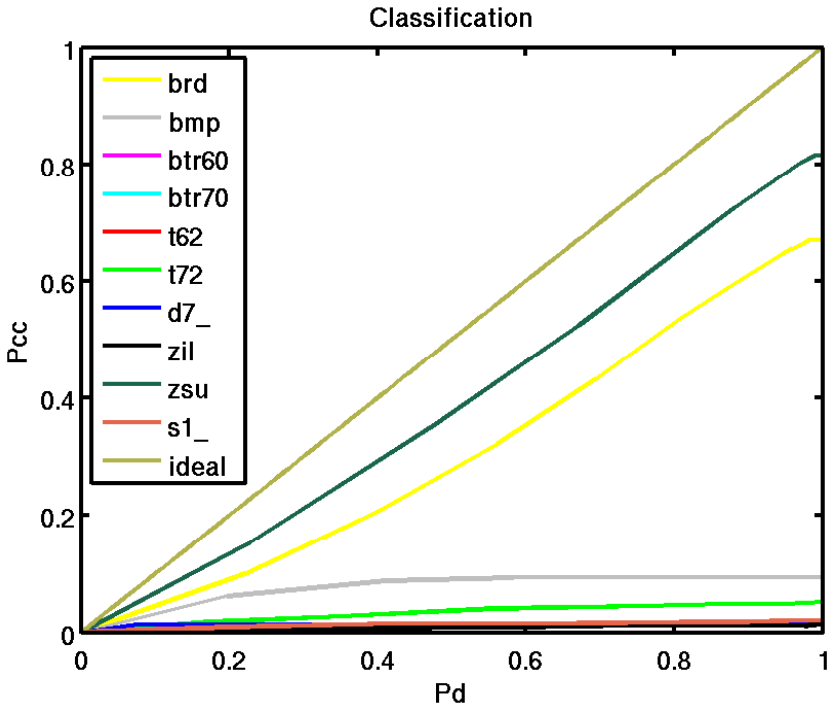
Figur 11.2: Eksempel på plott av deklarasjon mot falsk alarm for klassene i MSTAR-datasettet.

målinger fra alle klassene.

De ukjente målene kan være fra helt andre datasett og kan være av helt andre klasser enn de i treningssettet, eller man kan velge å bare trene klassifikatoren på enkelte av de klassene man har tilgjengelig i treningssettet og bruke resten som ukjente. Dette kan være med på å avgjøre om deklarasjonen virkelig er så god som man tror den er, for data fra andre datasett kan ofte være så forskjellige fra det man jobber med at det er lett å ikke deklarere dem i første omgang.

Terskelverdiene man skal teste med bør velges slik at man får gyldige verdier for P_{fa} i hele intervallet $[0, 1]$. Det finnes ulike måter denne terskel-testingen kan implementeres på. Man kan enten se på absoluttverdien av diskriminantfunksjonene eller se på hvor mye høyere enn de andre vinnerklassens diskriminantfunksjonen er.

For å enkelt kunne sammeligne resultater fra forskjellige tester ved å sette en verdi på ytelsen har vi valgt å bruke den største avstanden mellom gjenomsnittet av P_{cc} for alle klassene og P_{fa}. Dette ble valgt for å slippe å låse seg til en P_{fa}. Den største verdien som er mulig å oppnå er 1, og for å få dette må alle gyldige mål detekteres og klassifiseres riktig, samt ingen falske alarmer. Dette er dermed den ideelle klassifikatoren. 0 er laveste verdi og den fåes når sannsynligheten for å detektere og klassifisere riktig er lik sannsynligheten for



Figur 11.3: Eksempel på plott av korrekt klassifikasjon mot deteksjon for klassene i MSTAR-datasettet.

falsk alarm. Er verdien negativ, dvs. at det er mer sannsynlig med falsk alarm enn korrekt klassifikasjon, blir ikke resultatet engang lagret i databasen, da dette er dårligere enn å bruke en klassifikator som velger klasse tilfeldig.

11.2 Generaliseringsproblemer

Testdata blir ofte hentet inn under ideelle forhold der bakgrunnen er relativt homogen. Man har kontroll på alle parametre. I virkelige data er objektet man er interessert i ofte plassert i en variert bakgrunn, og gjerne forsøkt skjult under trær o.a. Aspektvinkelen til objektet er som regel også ukjent. Denne må estimeres, og da er det fint å ha testdata der disse parametrene allerede fins. På den måten kan man teste sine estimeringsmetoder opp mot en fasit. Man trenger derfor både testdata med fasit for utvikling og deretter virkelige data for å kunne vurdere ytelsen til systemet. Det er også en fordel å ha flere datasett å jobbe på for å kunne sjekke en klassifikator opp mot objekter av samme type, men som er samlet inn med et helt annet system.

	BRD	BMP	BTR	T62	T72	D7	ZIL	ZSU	2S1	ukjent	%
BRD	93	0	0	2	0	0	3	0	2	10	93.0
BMP	1	87	2	0	0	0	2	0	0	18	94.6
BTR	3	0	80	1	1	0	2	6	2	15	84.2
T62	4	3	2	91	0	0	1	1	0	8	89.2
T72	7	3	4	1	84	0	0	0	0	12	84.8
D7	10	0	2	0	0	87	0	2	0	9	86.1
ZIL	4	1	2	0	0	0	94	0	0	9	93.1
ZSU	3	1	3	1	0	0	0	89	0	9	93.1
2S1	3	1	7	2	1	0	0	5	84	7	81.6
2S1(uavh.)	2	2	5	0	4	2	9	4	66	12	70.2
Ukjent1	0	0	4	2	8	2	0	5	7	83	74.7
Ukjent2	1	5	7	3	8	1	4	0	4	79	70.5

Tabell 11.1: Eksempel på forvirringsmatrise for MSTAR-datasettet

12 Resultater

Vi har implementert de tre klassifikatorene som er beskrevet i kapittel 10. MSTAR-datasettet er brukt i alle testene. Her er objektene allerede detektert, dvs. at hvert bilde kun inneholder ett objekt med litt bakgrunn rundt. Bildene blir segmentert og så sendt rett inn i klassifikatoren uten preprosessering.

Et problem med disse testene er at vi ikke har SAR-bilder av de samme målene som i treningssettet fra andre sensorer og vi får dermed ikke med de uavhengige klassene som vist i tabell 11.1. Den ukjente klassen i disse testene inneholder bilder av tilfeldige objekter fra Sandia-datasettet. I denne klassen burde vi ideelt sett hatt et mye bredere utvalg av SAR-bilder fra forskjellige sensorer som inneholder mange ulike objekter, gjerne med relativt lik størrelse som objektene i treningsettet. Vi må derfor jobbe videre med å lage en bedre ukjent-klasse.

12.1 k-nærmeste nabo

Metoden er implementert som forklart i kapittel 10.4. Mange forskjellige kombinasjoner av egenskaper ble testet, og disse ga best resultatet for denne klassifikatoren:

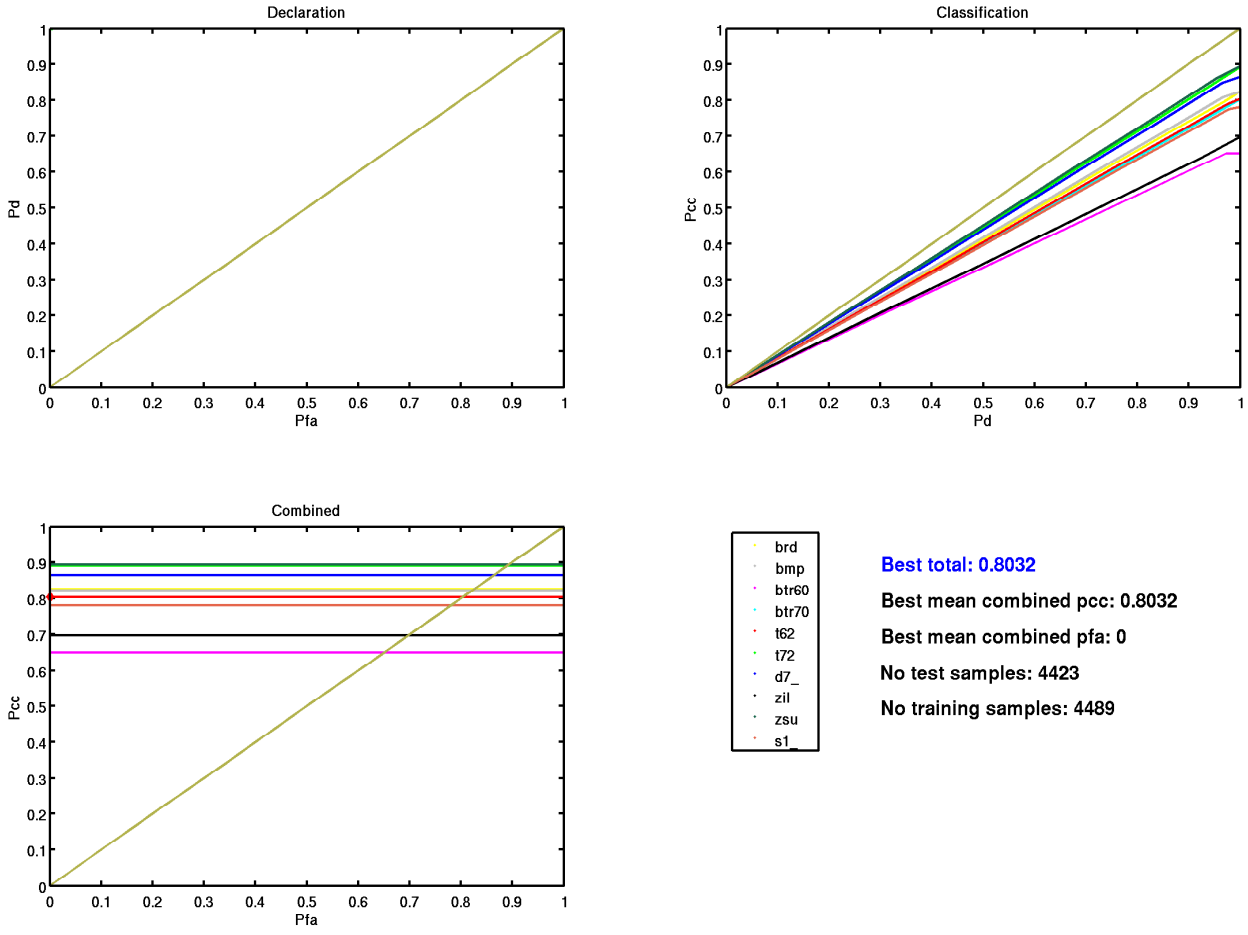
- Gjennomsnittlig pikselverdi

- Tre forskjellige lengdeestimater
- Eksentrisitet
- Euler-nummer
- Utstrekning
- Soliditet
- Lengde, bredde og diameter til rektangelet som best omslutter de segmenterte pikslene
- Fire forskjellige breddeestimater
- Antall piksler i den konvekse omslutningen
- Dynamisk område
- Sterkeste spredere

For å teste terskelverdier har vi sett på hvor langt unna sample k er. Beste k viste seg å være 3. I denne testen er det bare sammenlignet med treningsdata fra et vinkelspenn på 10 grader rundt det aktuelle testsampelet. Resultatene er vist i figur 12.1. Vi ser at deklarasjonen er god, mens klassifikasjonen kunne vært bedre. Grunnen til at deklarasjonen ble såpass god er at objektene i ukjent-klassen ikke er tilstrekkelig like dem i treningssettet. Grunnen til at klassifikasjonen ikke ble bedre er at det i databasen sannsynligvis ikke finnes en kombinasjon av egenskaper som klarer å skille mellom klassene, og vi må jobbe videre for å finne bedre egenskaper.

Tabell 12.1 viser forvirringsmatrisen i det punktet der gjennomsnittet av $P_{cc} - P_{fa}$ er høyest. Vi har kjørt noen tester for å se hvordan dette resultatet endres ved å variere vinkelspennet som brukes. Figur 12.2 viser et plott av ytelse mot størrelsen på vinkelspennet med de samme parametrene. Bare vinkelspennet endres. Effekten dette steget vil ha på ytelsen til klassifikatoren vil selvfølgelig avhenge av nøyaktigheten til vinkelestimatet. Vi ser at et vinkelspenn på 10 grader ser ut til å være optimalt for disse parametrene.

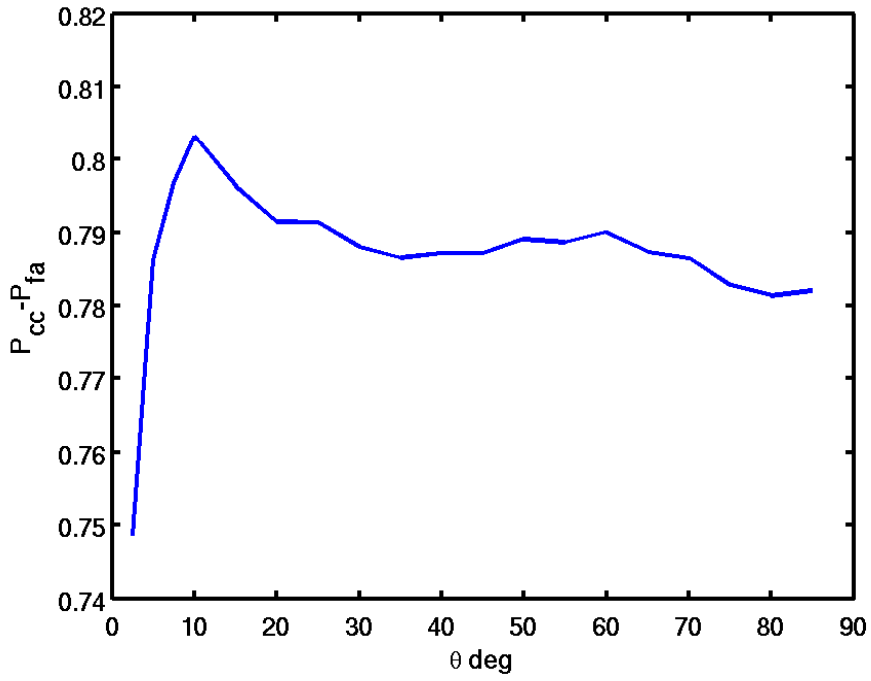
I figur 12.3 varieres bare k, mens alle de andre parametrene holdes faste. k er antallet naboer som testsampelet sammenlignes med. I dette tilfellet ble resultatet best ved å se på de 3 nærmeste naboene. Dette tallet vil være avhengig av egenskapene som brukes og hvor tett tilstandsrommet er samplet.



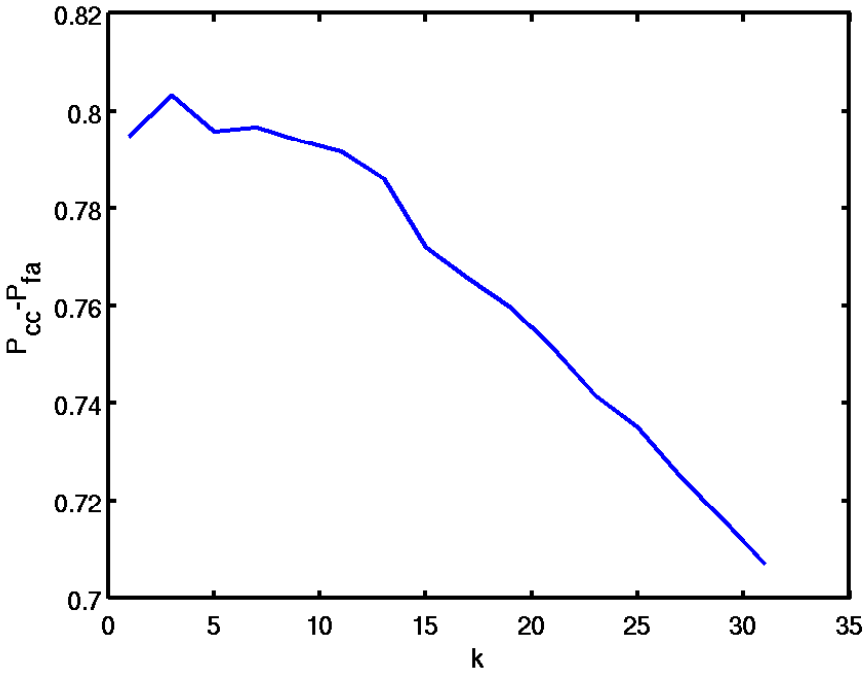
Figur 12.1: ROC-kurver for k-nærmeste nabo-metoden. Det er vanskelig å se noe i det første plottet, men her ble alle objektene deklarert, og det var ingen falske alarmer.

	brd	bmp	btr60	btr70	t62	t72	d7	zil	zsu	s1	ukjent	%
brd	566	41	15	8	1	4	0	4	31	16	0	82.51
bmp	40	527	16	12	1	11	1	3	16	14	0	82.22
btr60	19	25	164	23	0	2	0	3	4	12	0	65.08
btr70	5	13	9	167	0	0	0	0	1	13	0	80.29
t62	0	0	0	0	242	38	4	6	9	2	0	80.40
t72	1	30	3	0	13	582	4	4	5	12	0	88.99
d7	0	1	0	0	6	4	229	10	15	0	0	86.42
zil	1	4	2	1	18	22	5	192	11	19	0	69.82
zsu	14	20	0	0	16	8	10	4	605	0	0	89.36
s1	11	21	12	24	6	11	0	5	8	350	0	78.12
xxx	0	0	0	0	0	0	0	0	0	0	82	100.00

Tabell 12.1: Forvirringsmatrise for k-nærmeste nabo-metoden



Figur 12.2: Forskjellige vinkelspenn med k-nærmeste nabo-metoden. Alle andre parametre er faste. 10 grader ser ut til å gi det beste resultatet.

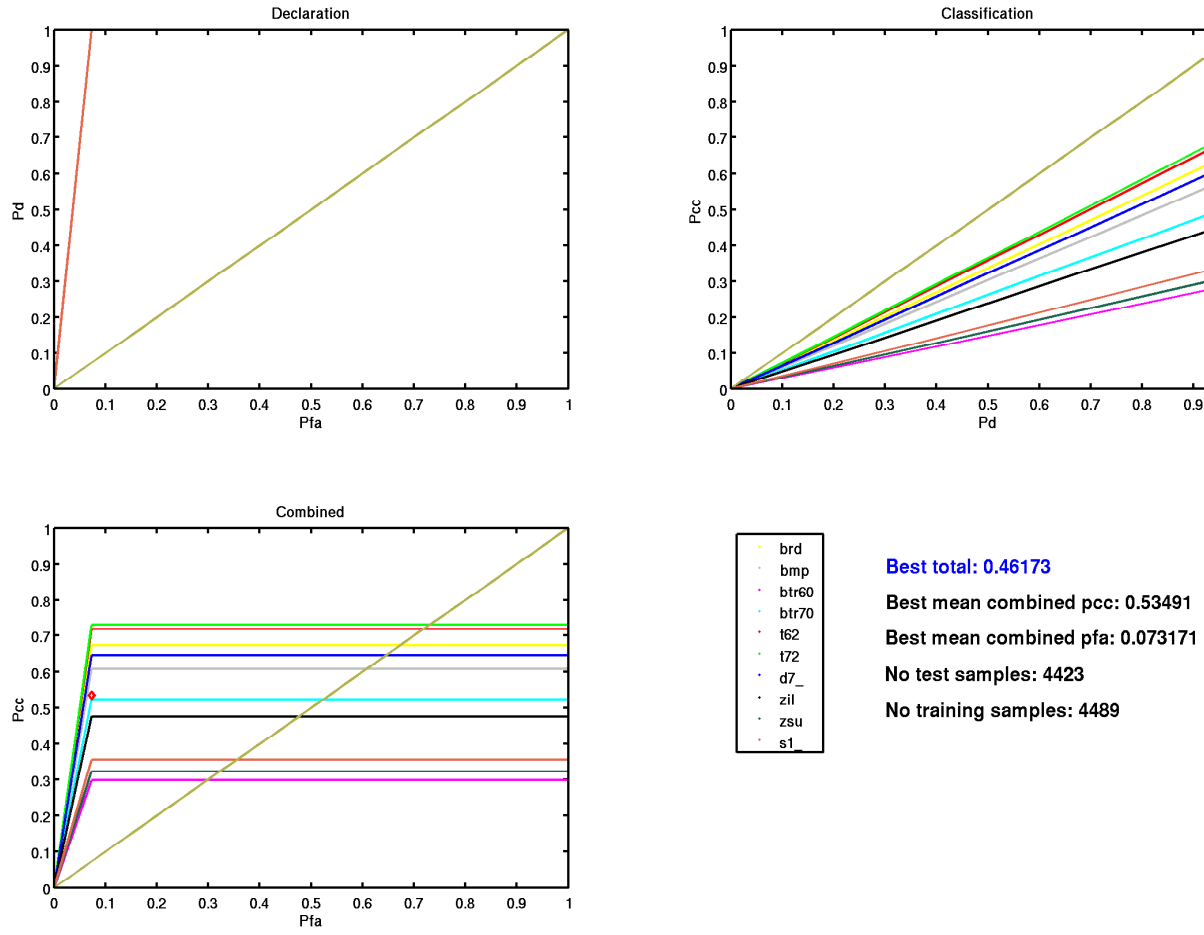


Figur 12.3: Forskjellige k med k-nærmeste nabo-metoden. Alle andre parametre er faste. 3 naboer ser ut til å gi best resultat.

12.2 Minimum feilrate

Vi har implementert en versjon av minimum feilrate-klassifikatoren som benytter den symmetriske feilfunksjonen som omtalt i kapittel 10.3. Mange forskjellige kombinasjoner av egenskaper ble testet, og disse ga best resultatet for denne klassifikatoren:

- Standardavvik imaginærdel
- Kurtosis
- Kurtosis imaginærdel
- Klippet gjennomsnitt
- Avstand mellom noen spredere
- Et breddeestimat
- Bredden til rektangelet som best omslutter de segmenterte pikslene



Figur 12.4: ROC-kurver for minimum feilrate-metoden

De underliggende fordelingene er antatt normale. Dette er sannsynligvis ikke en gyldig antakelse og figur 12.4 viser at dette heller ikke er tilfelle. Parametrene til fordelingen ble estimert med ML-metoden. Hadde andre fordelinger blitt antatt hadde metoden muligens virket bedre. Evt. er det mulig å kjøre en statistisk analyse av egenskapene for å finne ut om de kan ha blitt generert av en annen kjent fordeling. Tabell 12.2 viser forvirringsmatrisen i det punktet der gjennomsnittet av $P_{cc} - P_{fa}$ er høyest.

12.3 Neuralt nett

Nettet som ble testet er et tilbakepropageringsnett som omtalt i kapittel 10.5. Mange forskjellige kombinasjoner av egenskaper ble testet, og disse fikk best resultat med denne

	brd	bmp	btr60	btr70	t62	t72	d7	zil	zsu	s1	ukjent	%
brd	462	42	16	76	2	1	0	69	8	10	0	67.35
bmp	27	389	35	48	43	10	1	23	29	36	0	60.69
btr60	5	25	75	70	3	1	0	14	17	42	0	29.76
btr70	0	16	45	109	0	0	0	7	6	25	0	52.40
t62	0	8	0	2	216	53	3	2	13	4	0	71.76
t72	0	30	2	1	127	478	5	1	9	1	0	73.09
d7	0	3	0	0	27	50	171	7	7	0	0	64.53
zil	2	29	9	19	37	7	1	131	24	16	0	47.64
zsu	8	233	17	7	38	25	4	6	218	121	0	32.20
s1	8	81	59	60	31	3	0	11	36	159	0	35.49
xxx	2	0	0	0	0	0	3	0	0	1	76	92.68

Tabell 12.2: Forvirringsmatrise for minimum feilrate-metoden

klassifikatoren:

- Gjennomsnitt
- Masse
- Harmonisk gjennomsnitt
- To breddeestimater
- Avstand mellom noen spredere
- Dynamisk område
- Et lengdeestimat
- Lengden til rektangelet som best omslutter de segmenterte pikslene
- Styrkeforhold mellom spredere

Heller ikke denne klassifikatoren gjorde det like bra som nærmeste nabo, noe som betyr at egenskapene ikke separerer godt nok mellom de ulike klassene, og vi må prøve å finne noen bedre. I slike tilbakepropageringsnett er det mange parametre å sette. Andre valg kunne muligens gitt bedre restultater, men sannsynligvis er det egenskapene selv som ikke er gode nok. Tabell 12.3 viser forvirringsmatrisen i det punktet der gjennomsnittet av $P_{cc} - P_{fa}$ er høyest.

	brd	bmp	btr60	btr70	t62	t72	d7	zil	zsu	s1	ukjent	%
brd	563	20	2	3	0	2	0	16	71	8	1	82.07
bmp	102	403	6	0	10	15	1	7	73	24	0	62.87
btr60	44	15	82	22	0	2	0	15	19	53	0	32.54
btr70	22	6	29	98	0	2	0	14	5	32	0	47.12
t62	3	17	1	0	206	57	7	2	4	4	0	68.44
t72	1	39	0	0	28	568	7	0	6	5	0	86.85
d7	2	4	4	3	8	14	221	2	5	1	1	83.40
zil	8	17	5	5	17	18	7	141	9	48	0	51.27
zsu	29	9	4	0	35	21	13	3	556	7	0	82.13
s1	20	48	12	3	7	18	0	19	9	310	2	69.20
xxx	0	0	0	0	0	0	0	0	0	0	82	100.00

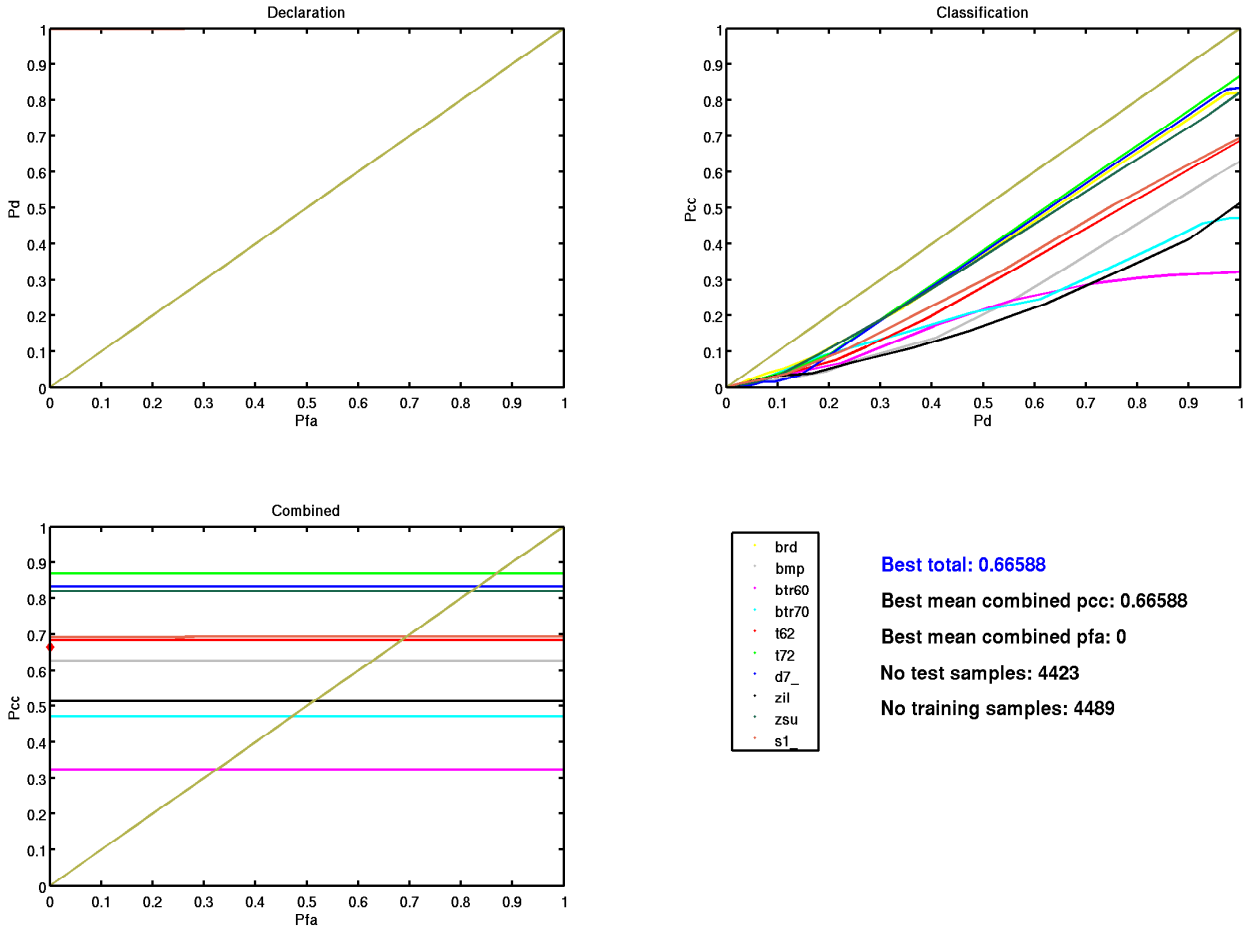
Tabell 12.3: Forvirringsmatrise for neuralt nettverk

Det var ikke de samme egenskapene som ga det beste resultatet for de tre klassifikatorene. En mulig forklaring på dette er at ingen av egenskapene i databasen (eller kombinasjoner av disse) var gode nok til å skille ordentlig mellom de ulike klassene. De forskjellige klassifikatorene vektlegger ulike ting og har ulike feilfunksjoner, og hvis ingen kombinasjoner av egenskaper er optimale, vil de ikke gi likt resultat selv med de samme kombinasjonene.

13 Konklusjon

Denne rapporten har beskrevet arbeid som er blitt gjort innen SAR ATR på ugraderte data i prosjekt SOBEK. Rapporten beskriver en del teori og noen resultater. Tre klassifikatorer er blitt implementert, k-nærmeste nabo, minimum feilrate-metoden og et neuralt nettverk. Mye arbeid er lagt ned i å strukturere data, både egenskaper fra SAR-bilder og resultater av tester, i databaser. Før objekter i et SAR-bilde kan klassifiseres må bildet gjennom deteksjon, segmentering og eventuelt annen preprosessering. Disse temaene har blitt beskrevet.

Testene vi har gjort av de ulike klassifikatorene viser at blandt de tre vi har implementert er det k-nærmeste nabo som gjør det best. Denne er imidlertid også den tregeste. For hver klassifikator ble et stort antall ulike kombinasjoner av egenskaper testet. Av de egenskapene som lå i databasen var det ulike kombinasjoner som ga best resultat for de tre klassifikatorene. En mulig forklaring på dette er at ingen av egenskapene i databasen (eller kombinasjoner av disse) var gode nok til å skille ordentlig mellom de ulike klassene. De forskjellige klassifikatorene vektlegger ulike ting, og hvis ingen kombinasjoner er optima-



Figur 12.5: ROC-kurver for neuralt nettverk. Det er vanskelig å se noe i det første plottet, men her ble alle objektene deklarert, og det var ingen falske alarmer.

le, vil de ikke gi likt resultat med de samme kombinasjonene av egenskaper. Utfordringen med å designe en klassifikator blir dermed å finne egenskaper som er robuste, dvs. holder seg stabile selv om avbildningsgeometrien forandres, og som kan skille mellom klassene i treningssettet. Med de eksisterende egenskapene i databasen ville sannsynligvis heller ikke andre klassifikatorer fått gode resultater, men vi skal likevel prøve å implementere flere i senere arbeider. Det viktigste blir likevel å lete etter gode egenskaper, noe som krever innsikt i hvordan SAR-bilder dannes, og hvordan ulike objekter vil opptre i bildene.

Det er nyttig å kunne sammenligne ytelsen til en klassifikator med andre. En måte å gjøre dette på er blitt beskrevet.

Felles for alle klassifikatorene er at ytelsen blir bedre hvis man legger inn korrekt aspektvinkel. For å få et godt estimat av denne har vi presentert en måte å fusjonere ulike estimater på ved hjelp av et neuralt nettverk. Det viste seg at denne metoden alltid var bedre enn å bruke de individuelle estimatene. Samme metode kan brukes for å estimere objektets lengde og bredde. Virkelig lengde og bredde for målene i MSTAR-datasettet er likvel så like at det er lite sannsynlig at man vil klare å estimere dem nøyaktig nok til å skille mellom klassene bare basert på disse estimatene. I datasett der målene er av mer variabel størrelse vil disse estimatene være viktigere.

Testene i denne rapporten har vært gjort på MSTAR-datasettet, et datasett som er laget spesielt for ATR. Man må ta hensyn til andre faktorer hvis man skal jobbe med flere datasett og data fra andre sensorer. Men som et første steg på veien er det enkelt å bruke data der man har fasiten og informasjon om de forskjellige objektene. Det er derimot lang vei fra å kunne klassifisere MSTAR-data til operativ bruk av en eventuell klassifikator. Da må man også jobbe mer med de første stegene i prosesseringskjeden, nemlig robust deteksjon og segmentering i ujevne bakgrunner. En deteksjonsalgoritme som klarer å plukke ut bare de interessante objektene i store datamangder vil være tidsbesparende for operatøren.

Videre i SOBEK vil fokuset være på å prøve å utnytte de komplekse bildene, istedenfor å bare se på intensitetsbildene, for å finne robuste egenskaper som kan skille bedre mellom de ulike klassene av mål. I tillegg skal vi se mer på hvordan vi kan preprosessere bildene for å best få frem informasjonen i dem som kan diskriminere mellom klassene.

Referanser

[1] SET-053, "Ground target automatic recognition by radar [NATO RESTRICTED]," RTO Technical Report RTO-TR-IST-999, NATO Reasearch and Technology Organisation, February 2008.

[2] A. O. Knapskog, T. Sparr, and S.-E. Hamran, "SAR/ISAR og MTI mot overflatemål - En gjennomgang av teknologi og operativ nytteverdi," FFI/RAPPORT 2003/00813, Forsvarets Forskningsinstitutt, 2003.

[3] C. Cochin, P. Pouliguen, B. Delahaye, D. Le Hellard, P. Gosselin, and F. Aubineau, "MOCEM - An 'all in one' tool to simulate SAR image," *Proceedings of the Seventh European Conference on Synthetic Aperture Radar*, vol. 1, pp. 225–228, June 2008.

[4] A. W. Rihaczek and S. J. Hershkowitz, *Theory and practice of radar target identification*. Artech House, 2000.

[5] C. Oliver and S. Quegan, *Understanding Synthetic Aperture Radar Images*. Artech House, 1998.

[6] M. di Bisceglie and C. Galdi, "CFAR detection of extended objects in high-resolution SAR images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 43, pp. 833–843, April 2005.

[7] M. Guida, M. Longo, and M. Lops, "Biparametric linear estimation for CFAR against Weibull clutter," *IEEEAE*, vol. 28, pp. 138–151, January 1992.

[8] S. Kuttikkad and R. Chellappa, "Non-gaussian CFAR techniques for target detection in high resolution SAR images," *Proceedings of the IEEE International Conference on Image Processing*, vol. 1, pp. 910–914, November 1994.

[9] Z. Bi, J. Li, and Z. Liu, "Super resolution SAR imaging via parametric spectral estimation methods," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 35, no. 1, pp. 267–281, 1999.

[10] J. Li and P. Stioca, "Efficient mixed-spectrum estimation with applications to target feature extraction," *IEEE Transactions on Signal Processing*, vol. 44, no. 2, pp. 281–295, 1996.

[11] J. Tsao and B. D. Steinberg, "Reduction of sidelobe and speckle artifacts in microwave imaging: The CLEAN technique," *IEEE Transactions on Antennas and Progagation*, vol. 36, no. 4, pp. 543–556, 1988.

[12] H. Callow, J. Groen, R. E. Hansen, and T. Sparr, “Shadow enhancement in SAR imagery,” *Proceedings of the IET International Conference on Radar Systems*, October 2007.

[13] X. Liao, Z. Bao, and M. Xing, “On the aspect sensitivity of high resolution range profiles and its reduction methods,” *IEEE International Radar Conference*, pp. 310–315, 2000.

[14] X. Liao, P. Runkle, and L. Carin, “Identification of ground targets from sequential high-range-resolution radar signatures,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, pp. 1230–1242, October 2002.

[15] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. John Wiley Sons, INC, 2001.

[16] O. Hellwich, “Line extraction from synthetic aperture radar scenes using a Markov random field model,” *Proceedings of SPIE*, vol. 2958, pp. 107–113, 1996.

[17] N. M. Sandirasegaram, “Automatic target recognition in SAR imagery using a MLP neural network,” Technical memorandum TM 2002-120, DRDC Ottawa, November 2002.