Universitetet
i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

MASTEROPPGAVE

Studieprogram/spesialisering: Informasjonsteknologi, kybernetikk/signalbehandling	Vår semesteret, 2009 Åpen / Konfidensiell
Forfatter: Gaute Dirdal Lunde	…… (signatur forfatter)
Faglig ansvarlig: Trygve Eftestøl (UiS) Veileder(e): Trygve Eftestøl (UiS), Torodd Buer (Davo AS)	
Tittel på masteroppgaven: Bruk av nevrale nettverk for avansert deteksjon av objekter under vann. Engelsk tittel:	
Studiepoeng: 30	
Emneord: SwimEye Nevrale nettverk	Sidetall: 30 + vedlegg/annet: 27 Stavanger, 12.06.2009 dato/år

Sammendrag

SwimEye er et system for å øke sikkerheten mot drukningsulykker i basseng. Det er ønskelig å gjøre SwimEyes deteksjon enda bedre, og denne masteroppgaven kommer frem til en løsning på dette problemet ved bruk av et nevralt nettverk.

Dataene som skal behandles her er binære bilder (bilder bestående av kun sorte eller hvite piksler) som er hentet ut fra SwimEye. Ulike nevrale nettverksstrukturer ble implementert i Matlab, og bedømt ut fra følgende kriterier:

- Sannsynlighet for korrekt klassifisering.
- Sensitivitet og spesifisitet.
- Areal under ROC kurvene.

Det ble funnet ut at et nevralt nettverk med ett skjult lag bestående av fem skjulte noder egner seg best. Dette nettverket har en sannsynlighet for korrekt deteksjon på 80,4 % i gjennomsnitt. En får da redusert andel falske positive alarmer med 58,4 % ved en sensitivitet på 95 %.

Selve programmet er implementert i Matlab og en beskrivelse av programmets funksjoner samt programkoden er gitt som vedlegg.

Forord

Jeg vil benytte anledningen til å takke mine to veiledere for denne hovedoppgaven, Torodd Buer og Trygve Eftestøl. Spesielt vil jeg takke Trygve som har latt kontordøren stå åpen slik at jeg kunne komme inn for å diskutere oppgaven. Jeg vil også takke de søte jentene i bokkaffen på UIS som alltid har laget god kaffe til meg, det har blitt en del kopper i løpet av dette semesteret.

Forkortelser

Følgende forkortelser er brukt i denne rapporten:

Receiver Operating Characteristics, ROC.

Sanne negative, TN .

Sanne positive, TP .

Mean Square Error, MSE.

Notasjon

Følgende notasjon er brukt på matematiske størrelser i denne rapporten:

Vektorer skives med små bokstaver med pil over, eks $\vec{x}$.

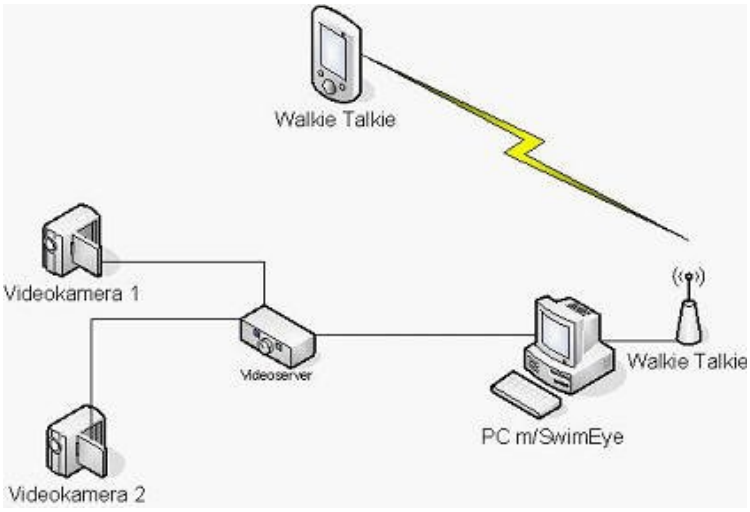
Matriser skrives med store bokstaver med understrek, eks $\underline{X}$.

Skalarer skrives som enten store eller små bokstaver, eks X eller x.

Innholdsfortegnelse

1 Innledning

SwimEye [1] er et produkt har som oppgave å detektere mennesker som kan være i fare for å drukne i basseng. De viktigste komponentene som inngår i SwimEye er videokameraer, videoserver, PC med SwimEye og en Walkie Talkie (se figur 1.1).



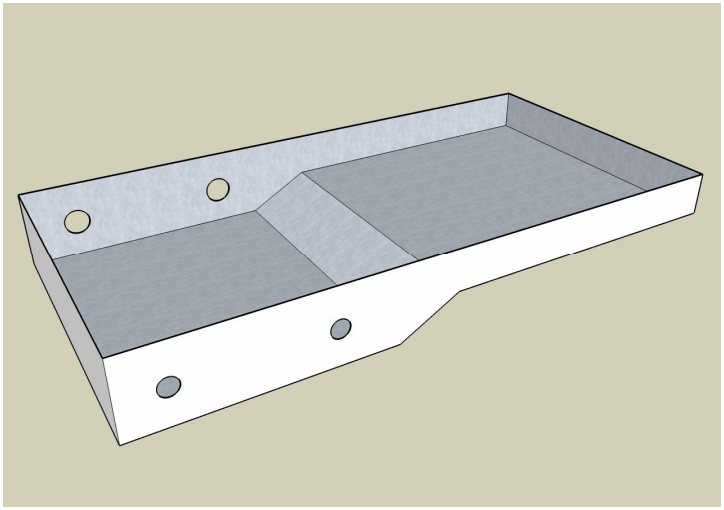
Figur 1.1: Prinsippskisse for SwimEye [1].

Fra et eller flere videokameraer nede på veggen i bassenget får SwimEye inn bilder fra bassengbunnen (se figur 1.2). De runde hullene på figuren er kamerahus hvor kameraene plasseres. Bildene blir analysert i en datamaskin for å detektere eventuelle drukningsulykker. Personer som er for lenge under vann blir registrert og en alarm sendes da til badevakten.

Davo AS [1] er selskapet som innehar og forvalter alle rettigheter i forhold til SwimEye. Selskapet ble startet 15. november 2005 av Torodd Buer, Oral Sjøflot og Arne Martin Meling. Den første versjonen av SwimEye, kalt SwimEye V1 ble klar sommeren 2006 etter en seks måneders lang uttestingsperiode i Stavanger Svømmehall.

Davo AS jobber ut i fra et mål om at en falsk alarm pr dag er akseptabelt. Som alle vet er falske alarmer er svært uønsket og kan i verste fall føre til liknende situasjon som gutten som ropte ulv, ulv [2]. Tilliten til systemet kan altså bli svekket. I de fleste tilfeller klarer SwimEye

målsetningen på en falsk alarm pr dag, men ikke alltid. Denne oppgaven har derfor som mål å lage et program som kan hjelpe SwimEye med å raffinere detekteringen slik at den blir enda bedre. Det som er vektlagt i oppgaven er å skille mennesker/faste objekter fra støy. Med støy i denne sammenheng menes luftbobler, gjenspeiling av bassengbunnen og liknende.



Figur 1.2: SwimEye i basseng [1]

I siste avsnitt i kapittel 1 gis problemdefinisjonen, og videre i kapittel 2 beskrives bakgrunnen for problemet og en skisse av løsningen ved bruk av nevrale nettverk kommer frem. I kapittel 2.1 kommer en beskrivelse av data og behandling av disse, en ser på to egenskapsbeskrivelser og motiverer for bruk av nevrale nettverk. Videre i kapittel 3 blir det en del teori om morfologi, antall skjulte noder, antall skjulte lag og trening. Teori om ROC og terskelverdier som brukes for å beregne TP og TN kommer også i kapittel 3. Eksperimentene og resultatene blir presentert i kapittel 4, diskusjon og konklusjon kommer henholdsvis i kapittel 5 og 6.

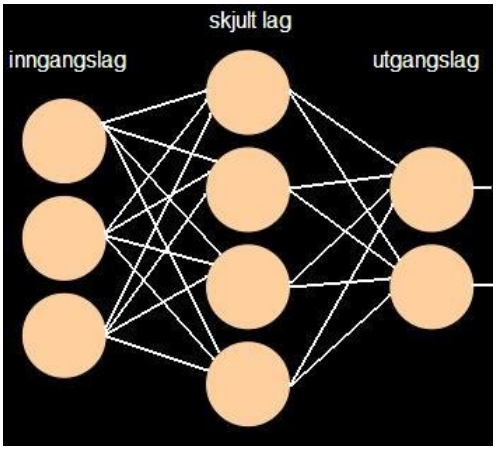
2 Bakgrunn og metoder

Dette kapittelet beskriver bakgrunnen for problemet og skisserer en løsningen ved bruk av nevrale nettverk. For å kunne trene opp et nevralt nettverk som skal kunne skille mennesker/faste objekter fra støy, må det trekkes ut noen egenskaper fra de faste objektene.

Det skal her ses på to slike egenskaper. Datainnsamling og behandling av data kommer også i dette kapittel og til slutt gis en motivasjon for hvorfor bruke nevrale nettverk.

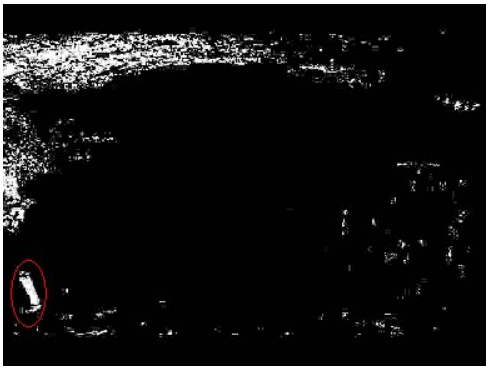
Dagens SwimEye teller opp antall hvite piksler i et binært bilde og klassifiserer objektet i bildet som menneske/fast hvis antall piksler er større enn en gitt grense. Dette er en meget enkel, men ikke så god metode å klassifisere på. Ved å bruke et nevralt nettverk vil kunne klare å skille mennesker/faste objekter fra støy bedre enn ved å bare telle opp antall hvite piksler.

Et nevralt nettverk er et nettverk av sammenkoblede noder. Disse nodene er forbundet med hverandre med ulike vekter, og representerer ulike lag i nettverket (inngangslag, skjult lag og utgangslag), se figur 2.1. Hver node har sin aktiveringsfunksjon som sørger for at utgangen på noden er innenfor et visst område vanligvis 0 og 1 eller -1 og 1. Nevrale nettverk er inspirert av biologiske nevrale nettverk i måten de behandler og prosesserer informasjon på.



Figur 2.1: Prinsippskisse for nevralt nettverk.

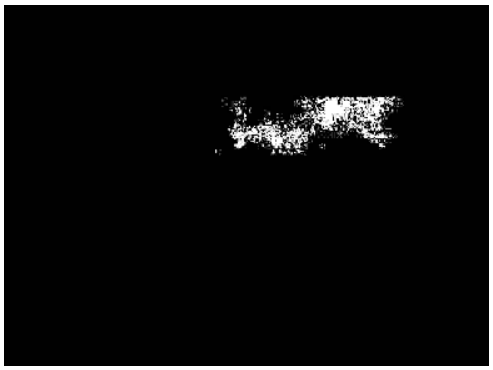
Figur 2.2: Analysebilde av menneske



Figur 2.3: Analysebilde av menneske og støy

På figur 2.2 ser en tydelig at det er et menneske som er i bildet, men på figur 2.3 kan en ikke umiddelbart se noe menneske, den ene foten er derfor markert med rød ring (det er faktisk bare denne ene foten som er med i bildet). Det andre en ser på figuren er støy, her i form av luftbobler.

Figur 2.4: Analysebilde av støy



Figur 2.5: Analysebilde av støy

Figur 2.4 viser ren støy i form av sollys som reflekteres av bassengbunnen. Figur 2.5 viser støy i form av luftbobler.

Denne rapporten skal vurdere hvor godt et nevralt nettverk lar seg bruke for å løse problemet. For å kunne gjøre dette må en først definere noen gode egenskaper som definerer et menneske/fast objekt. Etter at disse egenskapene er definert brukes de til å trene opp et nevralt nettverk slik at mennesker/faste objekter kan skilles fra støy.

2.1 Datainnsamling og behandling

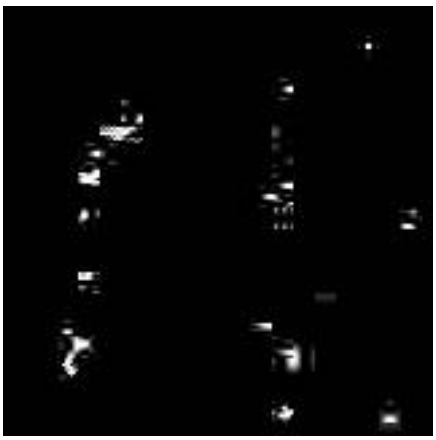
Det er blitt hentet ut 226 jpeg bilder fra SwimEye hvorav 126 av dem er definert som faste objekter, og 100 er definert som støy. Merk at alle disse bildene er reelle situasjoner hvor SwimEye har gitt en alarm. Figur 2.4 og figur 2.5 har altså utløs alarmer selv om dette bare er støy. Alle bildene er lagt inn i katalogen *analysebilder5*. Disse bildene er gråskalabilder og for at matlab skal kunne jobbe med dem, må de omdannes til binære bilder se figur 2.6 og figur 2.7. Dette gjøres i funksjonen *hentBilder* (se vedlegg A og B).

Figur 2.6: Gråskalabilde av fast objekt.

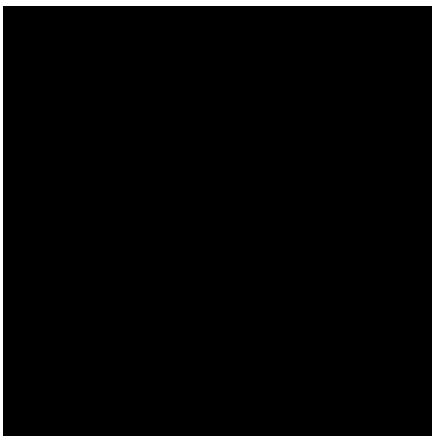
Figur 2.7: Binær bilde av fast objekt.

Den støyen som er enklest å skille fra de faste objektene er støyen vist i figur 2.4. Ved å gjøre en morfologisk åpning på bildene (se kapitel 3.1.3), vil mesteparten av slik støy nulles ut. Figur 2.8 og figur 2.9 viser en slik utnulling ved morfologisk åpning. Ved slike tilfeller vil de to egenskapene C og E (se kapittel 2.1.1) bli satt til null. Etter at de to egenskapene er trukket ut vil for eksempel det faste objektet i figur 2.7 være representer som

$$\vec{p} = \begin{bmatrix} egenskapC \\ egenskapE \end{bmatrix} = \begin{bmatrix} -0.458 \\ -0.913 \end{bmatrix} \quad (2.1)$$



Figur 2.8: Støy.



Figur 2.9: Støy etter morfologisk åpning.

2.1.1 Egenskapsbeskrivelse

Det å kunne beskrive noen egenskaper til objekter i bilder er veldig viktig med tanke på klassifisering. Målet til disse egenskapene er at de effektivt, unikt og mest mulig transformasjonsinvariant skal kunne representere viktige karakteristikker av et objekt, for eksempel i form av en vektor. Bildeinnholdet vil på denne måten forenkles og komprimeres.

Noen kriterier for god egenskapsbeskrivelse kan være:

- Kompakte egenskaper
- Lav beregningsmessig kompleksitet
- God nøyaktighet ved gjenskapning av objekter fra egenskapsbeskrivelsen

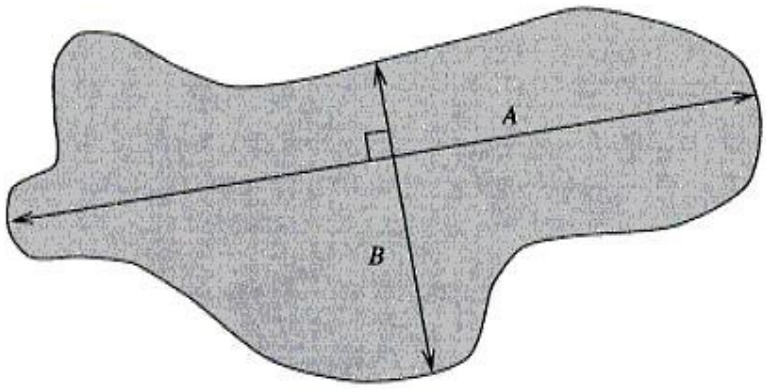
Egenskapsbeskrivelse kan deles inn i to hovedgrupper: konturbasert [3] og regionsbasert [3] Konturbasert egenskapsbeskrivelse betrakter bare konturen til objektene, dette har flere ulemper. Konturbasert egenskapsbeskrivelse er mer følsom for støy enn regionsbasert egenskapsbeskrivelse fordi objektenes innhold ikke tas med i betraktningen. I noen sammenhenger vil objektenes innhold være mer relevant enn objektenes kontur. Men det

finnes bruksområder hvor objektenes kontur er det viktigste og her vil konturbasert egenskapsbeskrivelse klart være å foretrekke.

Regionsbasert egenskapsbeskrivelse er mer robust siden den bruker all objektinformasjon. Disse egenskapsbeskrivelsene er generelt anvendelige og produserer også mer nøyaktige gjenskapelser. De kan også takle defekte konturer.

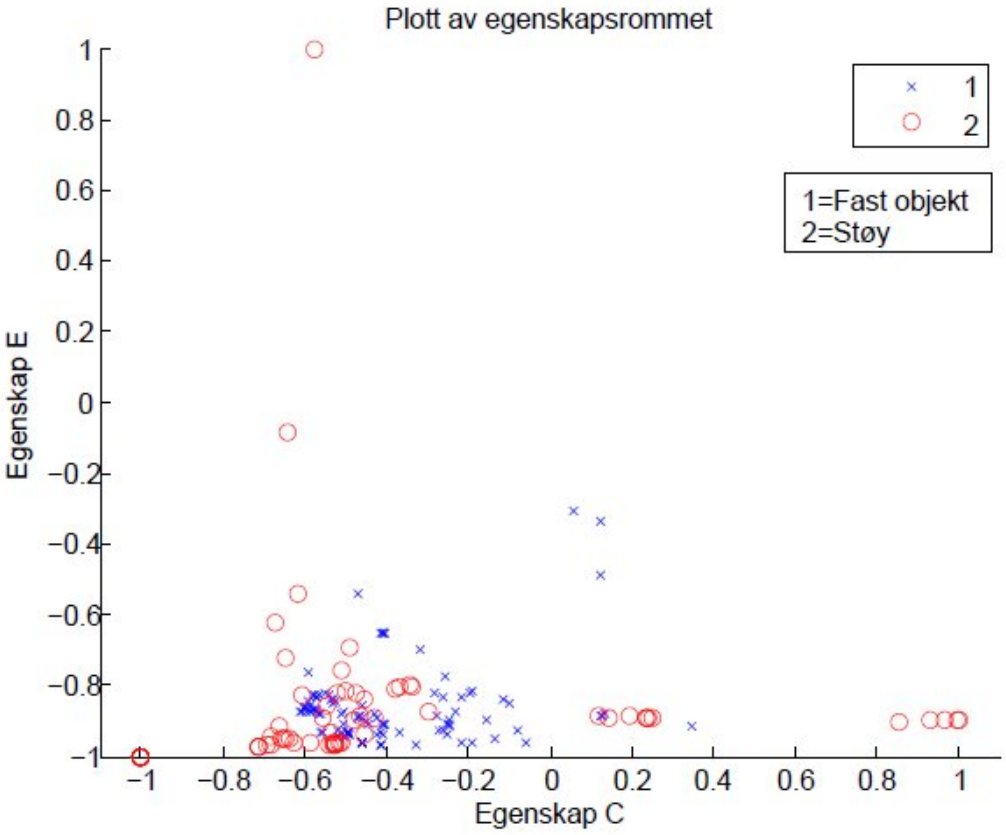
På grunn av at de faste objektene en i denne rapporten skal klassifisere varierer mye i form, så vil regionsbaserte egenskapsbeskrivelser egne seg best, og vil derfor bli anvendt.

Denne rapporten trekker ut to egenskaper fra bildene, egenskap C og E. Egenskap C beskriver gjennomsnittlig avstand mellom pikslene i bildet og egenskap E beskriver eksentrisiteten. Eksentrisitet [3] er forholdet mellom store og lille akse til objektet, se figur 2.10. Et fast objekt vil være større og mer kompakt (pikslene har mindre avstand mellom seg) enn det støyen er, derfor vil eksentrisitet og gjennomsnittlig pikselavstand egne seg godt til å beskrive mennesker/faste objekter.



Figur 2.10: Eksentrisitet [3].

For å få en føling med egenskapene, plottes egenskapsrommet til de to egenskapene (se figur 2.11). Merk at verdiene er normalisert slik at de er innenfor området [-1 1].



Figur 2.11: Plott av egenskapsrommet.

2.2 Hvorfor bruke nevrale nettverk

Dette delkapittelet er i stor grad hentet fra [4].

For å løse et problem med en vanlig datamaskin må maskinen ha et sett med klare definerte instruksjoner. Disse følges slavisk steg for steg og hvis neste steg for å løse problemet ikke er kjent for datamaskinen, vil den ikke komme videre. Nevrale nettverk lærer av eksempler og kan ikke bli programmert til å gjøre en bestemt oppgave.

De nevrale nettverkene har en veldig god evne til å utlede en mening fra veldig kompliserte datasett. Det nevrale nettverket vil kunne hente ut mønster og detektere trender i disse settene, noe som ville vert umulig for mennesker eller ved hjelp av vanlig

programmering. Som regel hvis en skal løse et problem trenger en informasjon om problemets statistiske fordeling, dette trenger en ikke hvis en bruker nevrale nettverk til å løse problemet. Men informasjon om statistisk fordeling vil kunne gjøre at treningen av nettverket går raskere. En kan tenke på det nevrale nettverket som en ekspert i den kategorien informasjon det har blitt trent opp til å analysere. Altså i vårt tilfelle en ekspert på å skille mennesker/faste objekter fra støy. Nevrale nettverk brukes ikke bare til å løse matematiske og vitenskapelige problemer men også problemer innen medisin, forretninger, finans og litteratur.

3 Teori

Som nevnt i kapittel 2.1 kan mesteparten av støyen i form av sollys som reflekteres av bassengbunnen fjernes ved å gjøre en morfologisk åpning på bildet. Morfologisk åpning [5] er en operasjon som består av en erosjon [5] etterfulgt av en dilasjon [5]. En kort innføring i disse tema gis i dette kapittel.

Å finne passende nettverksstruktur er viktig, antall skjulte noder og antall skjulte lag må bestemmes, dette er noe som ses på her. Etter at strukturen er funnet, må nettverket trenes opp og en ROC analyse kan deretter gjøres for å se hvor godt nettverket lar seg bruke til å klassifisere ut i fra de gitte egenskapene. Selve ROC analysen tas ikke i dette kapittel, men definisjonen, hva ROC er og hvordan en lager ROC kurver tas her.

3.1 Morfologisk filtrering

For å analysere og beskrive objekter i et bilde brukes matematisk morfologi. De fleste morfologiske operasjoner er beregnet for binære bilder, og mange av disse operasjonene brukes for å fjerne uønsket støy i bildet. Det meste av støyen i dette tilfellet, refleksjon av bassengbunnen (se figur 2.4), vil la seg fjerne fullstendig ved hjelp av morfologisk åpning. En

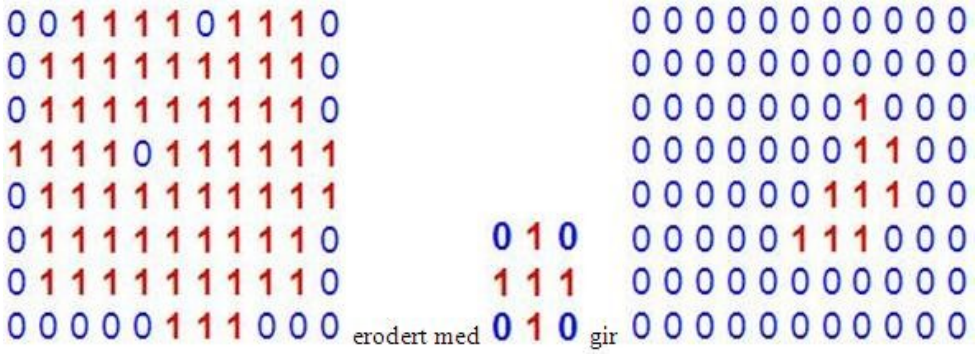
morfologisk åpning er en erosjon etterfulgt av en dilasjon. Figur 2.8 og figur 2.9 viser resultatet av en morfologisk åpning av støyen, som en ser vil støyen fjernes.

3.1.1 Erosjon

En erosjon av et binært bilde vil krympe et objekt i bildet. Erosjon av et sett A med et strukturelement B er definert som

$$A \Theta B = \{x \mid (B)_x \subseteq A\} \quad (3.1).$$

Dette vil si posisjonen til alle piksler x slik at B er inkludert i A når origo plasseres i x. Små utstikk på objektets omriss fjernes og innbuktninger på objektets omriss vil bli utvidet. Resultatet av erosjonen avhenger av størrelsen på strukturelementet, jo større strukturelement [5], jo mer erosjon.



Figur 3.1: Erosjon.

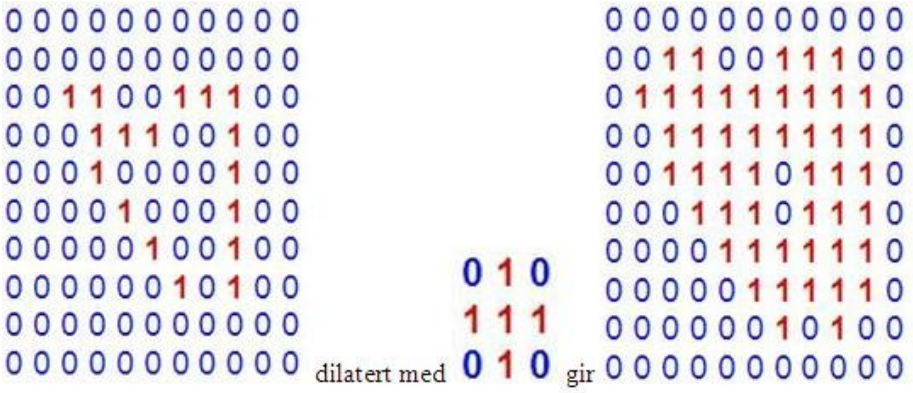
3.1.2 Dilasjon

I motsetning til erosjon vil en dilasjon av et binært bilde utvide objektet i bildet. En dilasjon av et sett A med et strukturelement B er definert som

$$A \oplus B = \{x \mid (\hat{B})_x \cap A \neq \emptyset\} \quad (3.2).$$

Altså posisjonen til alle piksler x slik at B overlapper med x i minst en piksel når origo plasseres i x. $\emptyset$ betegner den tomme mengde.

En dilasjon fyller hull i objektet og vil glatte ut innbuktninger i objektets omriss. Resultatet av dilasjonen avhenger også av strukturelementet, jo større strukturelement, jo mer dilasjon.



Figur 3.2: Dilasjon.

3.1.3 Morfologisk åpning

Hvis en utfører først en erosjon etterfulgt av en dilasjon med samme strukturelement, får en det som kalles en morfologisk åpning. En åpning av et sett A med et strukturelement B er definert som

$$A \circ B = (A \Theta B) \oplus B \quad (3.3).$$

Ved en morfologisk åpning vil de strukturene som ”overlevde” erosjonen bli gjenskapt. Denne operasjonen kan skape en åpning mellom to strukturer som kun henger sammen i en

tynn bro uten å krympe strukturene. I Matlab er det funksjonen *imopen* som utfører den morfologiske åpningen. Det er altså denne type morfologisk operasjon som vil fjerne mesteparten av den type støy som oppstår når sollys reflekteres av bassengbunnen (se figur 2.4).

3.2 Antall skjulte noder

Som nevnt tidligere består et nevralt nettverk av skjulte noder og skjulte lag.
Mens antall noder i inngangslaget er bestemt av dimensjonen på inngangsvektoren (her 2) og antall noder i utgangslaget er bestemt av antall klasser (her 2), så er det ikke noen åpenbar regel for antall noder i de skjulte lagene. n_H er ofte betegnelsen for antall skjulte noder og det er dette som regulerer uttrykkskraften til det nevrale nettverket, og dermed kompleksiteten til desisjonsgrensen [6]. Hvis en har lineært separable mønster som skal klassifiseres, så trenger en få skjulte noder, men hvis mønstrene kommer fra kompliserte tetthetsfunksjoner som er nøstet sammen, så trengs det flere noder. Uten mer informasjon så finnes det ingen generell metode for å sette antall skjulte noder før trening.

Med for mange skjulte noder risikerer man at nettet blir overtrent, det vil si at det bare fungerer på treningssettet og vil ikke fungere bra på nye data. Hvis en derimot har for få skjulte noder vil ikke nettverket ha nok frie parametere til å kunne tilpasse seg treningsdataene i treningssettet. I begge disse ekstreme senarier vil feilen når en kjører nettet på et uavhengig testsett bli uakseptabelt høy. En søker da et antall noder som gir lav testfeil.

Totalt antall vekter i nettet bestemmes av antall skjulte noder, dette kalles frihetsgrader. En bør av denne grunn ikke ha flere vekter enn det totale antall treningspunkt, n. En tommelregel er å sette antall skjulte noder slik at antall vekter blir ca $n/10$. En annen måte er å starte med et ”stort” antall skjulte noder og så redusere antallet etter hvert.

3.3 Antall skjulte lag

Bakoverforplantningsalgoritmen [6] fungerer like bra på nett med tre, fire eller flere skjulte lag, så lenge aktiveringsfunksjonene til nodene i disse lagene er deriverbare. Siden et trelags nevralt nett kan implementere enhver vilkårlig funksjon er det derfor som oftest å foretrekke. Hvis en har gitt problemspesifikke krav som må imøtekommes, kan det være aktuelt med flere enn tre skjulte lag. Eksempler på problemspesifikke krav er translasjon [5] og rotasjon [5].

3.4 Trening

For at det nevrale nettverket skal kunne gjenkjenne og klassifisere mennesker/faste objekter og støy, må det trenes opp. Treningsprotokoller brukes for å justere vektene slik at nettverkets uganger blir mer like målverdiene. Denne rapporten bruker programmerningsverktøyet Matlab til å implementere det nevrale nettverket. Matlab har en ”Neural Network Toolbox” med en rekke funksjoner for nevrale nettverk. Funksjonen *newff* genererer et bakoverforplantnings nevralt nettverk. Før treningen kan begynne må datasettet deles opp i treningssett, valideringssett og et uavhengig testsett, dette gjøres i funksjonen *dividevec*. Treningssettet brukes til å trene opp det nevrale nettverket mens valideringssettet brukes for å passe på at treningen stopper før nettverket blir overtrent. Blir nettverket overtrent vil det kun fungere bra på treningsdataene og vil ha en dårlig ytelse på nye data. Det uavhengige testsettet brukes for å gi en indikasjon på hvor godt nettverket vil fungere ute i felten. Matlab bruker treningsfunksjonen *trainlm* for trene det nevrale nettverket. Den oppdaterer vektene i henhold til Levenberg-Marquardts optimering [7]

$$w_{k+1} = w_k - \Delta w_k \quad (3.4)$$

$$\Delta w_k = [\underline{J(w_k)}^T * \underline{J(w_k)} + \mu * \underline{I}]^{-1} * \underline{J(w_k)} * \vec{e}(w_k) \quad (3.5).$$

Der $\underline{J(...)}$ er Jacobi matrisen [6], $\vec{e}(w_k)$ er feilvektoren, μ er læreraten [6] og $\underline{I}$ er identitetsmatrisen.

3.5 ROC teori

ROC kurver viser plot av sanne positive, TP, langs y-aksen og falske positive $(1-TN)$ langs x-aksen. Effektiviteten til klassifisereren (dens evne til å skille mennesker/faste objekter fra støy ut i fra de gitte egenskapene) måles i arealet under ROC kurven, stort areal tilsier god klassifiseringsevne, mens lite areal tilsier dårlig klassifiseringsevne. Dersom ROC kurven blir en diagonal linje fra (0,0) til (1,1) får vi arealet 0,5 og dette vil være det samme som å kaste mynt eller kron om klassifiseringen. Altså ønskes en ROC kurve som er på oversiden av diagonalen. Et ideelt resultat vil være et areal på 1.

Det å klassifisere et menneske som støy kan få katastrofale følger og må derfor undertrykkes så godt som mulig. Det nevrale nettverket baserer ytelsen kun på andel korrekte deteksjoner og tar ikke hensyn til alvorligheten ved de falske negative deteksjonene. Derfor må en sikre at TP raten blir høy. Dette gjøres ved å beregne noen terskelverdier, $\vec{\theta}$, (se kapitel 3.6) og videre gjøre en ROC analyse med de TP og TN ratene som er beregnet ut fra disse terskelverdiene.

Det er altså ikke nok i seg selv at nettverket har en høy sannsynlighet for korrekt deteksjon, det må også undertrykke de falske negative deteksjonene godt.

3.6 Terskelverdier

For å kunne gjøre en ROC analyse må en beregne en del terskelverdier. Når disse er beregnet kan en på bakgrunn av disse finne TP og TN.

Terskelverdien, θ, må bestemmes ut fra treningssettet. Først kjøres treningssettet gjennom det nevrale nettverket. Deretter hentes $g_1(\vec{p})$ og $g_2(\vec{p})$ verdiene ut. De to utgangene fra det nevrale nettverket, $g_1(\vec{p})$ og $g_2(\vec{p})$, kan sees på som aposteriori sannsynligheter for hver av klassene. En definerer

$$G = \frac{g_1(\vec{p})}{g_2(\vec{p})} \quad (3.6)$$

G sorteres så i synkende rekkefølge og tilhørende klassefasit finnes. For hver overgang mellom to klasser beregnes en terskelverdi. De to ytergrensene settes til

$$\theta_{start} = \max(G) + 4000 \quad (3.7)$$

og

$$\theta_{slutt} = \min(G) - 4000 \quad (3.8)$$

De resterende tersklene settes midt mellom overgangene mellom to klasser. For å gjøre dette litt klarere vises hvordan θ_2 beregnes. Tabell 3.1 viser et utsnitt av sorterte treningsdata og klassefasit. F betyr menneske/fast objekt, mens S betyr støy. En ser at første overgang mellom de to klassene skjer i linje 18 og 19 i tabell 3.1. θ_2 beregnes på følgende måte

$$\theta_2 = 13.460 + \frac{13.423 - 13.460}{2} = 13.442 \quad (3.9)$$

Nå kan en for hver terskelverdi beregne sann positiv og sann negativ rate på følgende måte

$$TP = \frac{antall\,\omega_1 > \theta_i}{\omega_{1,totalt}} \quad (3.10)$$

$$TN = \frac{antall\,\omega_2 < \theta_i}{\omega_{2,totalt}} \quad (3.11)$$

Der θ_i er terskel nr i, $\omega_{1,totalt}$ og $\omega_{2,totalt}$ er totalt antall faste objekter og støy henholdsvis.

Når dette er gjort beregnes TP og TN rater for testsettet og valideringssettet med de samme terskelverdiene, dette gjøres i *terskel8* (se vedlegg A og B).

1	Sorterte data	Klassefasit
2		
3	430,355	F
4	196,848	F
5	80,223	F
6	66,550	F
7	63,546	F
8	52,257	F
9	31,530	F
10	25,669	F
11	25,669	F
12	24,442	F
13	20,188	F
14	19,315	F
15	15,921	F
16	15,318	F
17	14,378	F
18	13,460	F
19	13,423	S
20	12,383	S
21	10,818	F
22	10,483	F
23	8,082	F
24	7,889	S

Tabell 3.1: Utsnitt av treningsdata.

4 Resultater

Nevrale nettverk med ulike strukturer har blitt implementert i Matlab og ytelsen analysert, alle er på formen 2-n_H-2. De ulike n_H som er testet ut i denne rapporten er *1, 3, 5, 7, 10, 15, 20, 30, 50*. Noen lærekurver presenteres samt ROC kurvene for det beste nettverket.

Rapporten har valgt å følge kapitel 3.2 sin anbefaling om et trelags nevralt nettverk. For å finne antall skjulte noder er følgende gjort:

Ulike nevrale nettverk har blitt trent opp 10 ganger hver og resultat har blitt beregnet (se tabell 4.1). Grunnen til at en trener nettverkene opp 10 ganger er for å få gjennomsnittlige verdier.

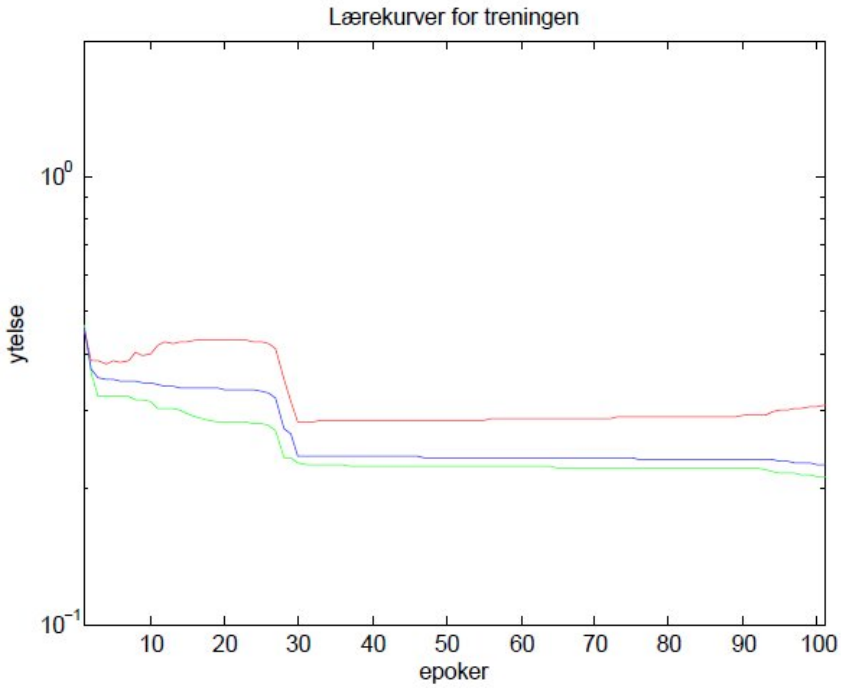
Antall skjulte noder	1	3	5	7	10	15	20	30	50
Gj.snitt korrekt klassifisering i testsett	0,572	0,682	0,804	0,427	0,867	0,828	0,833	0,552	0,525
Varians i testsett	0,001	0,011	0,005	0,005	0,004	0,002	0,001	0,002	0,011
Gj.snitt korrekt klassifisering i valideringssett	0,54	0,67	0,861	0,433	0,855	0,864	0,846	0,584	0,528
Varians i valideringssett	0,001	0,014	0,002	0,004	0,002	0,003	0,003	0,004	0,016
Gj.snitt korrekt klassifisering i treningssett	0,56	0,691	0,829	0,461	0,834	0,832	0,833	0,553	0,490
Varians i treningssett	0,001	0,011	0,005	0,005	0,004	0,002	0,001	0,002	0,011
Areal under ROC kurve for testsettet	0	0,712	0,846	0,499	0,812	0,782	0,807	0,705	0,500
Gj.snitt antall treningsepoker	0,1	20,3	8,6	11,3	9,5	10,3	13,1	18,9	12,3

Tabell 4.1: Resultater.

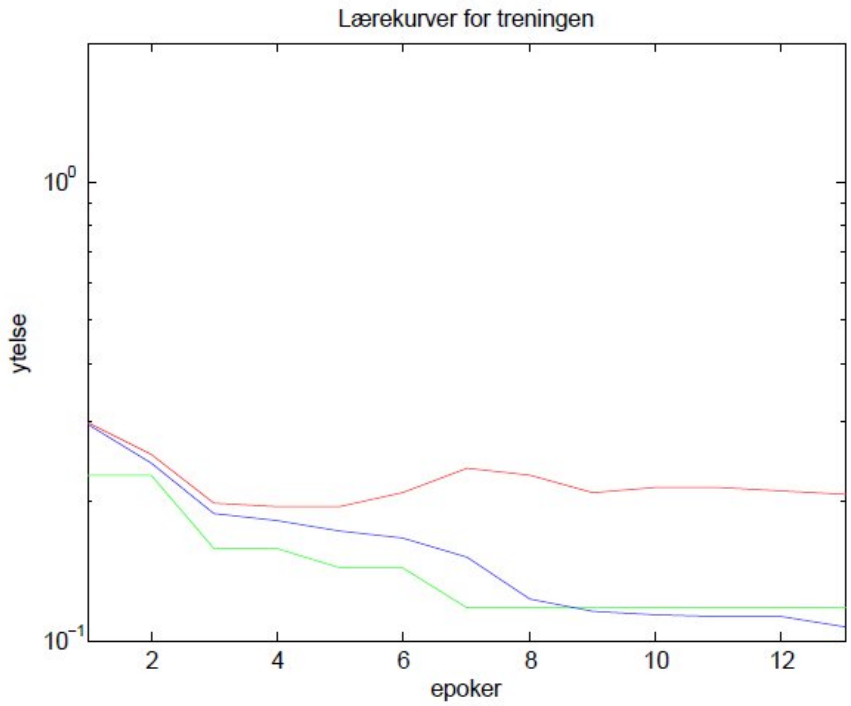
En ser tydelig at 1, 3, 7, 30 og 50 skjulte noder har vesentlig lavere sannsynlighet for korrekt klassifisering enn de resterende. En kan se ut ifra arealet under ROC kurven og den lave sannsynligheten for korrekt klassifikasjon at 1, 7 og 50 skjulte noder ikke egner seg til å skille mennesker/faste objekter fra støy. Med varians menes variansen i sannsynligheten for korrekt klassifikasjon. Det er lav varians for alle strukturene, dette tyder på at alle har god generalitet.

Lærekurvene viser ytelsen til nettverket. Ytelsen er basert på MSE og derfor vil et nettverk med ytelse nær null være å foretrekke siden målet er en MSE på null. De lærekurvene som vises i rapporten er for nettverk med *3, 5, 20,* og *30* skjulte noder. Blå kurve er fra treningen, grønn kurve er fra valideringen og rød kurve er fra testingen. Trenden er at ytelsen forbedres når antall skjulte noder økes (se figur 4.1- figur 4.3), men når n_H kommer

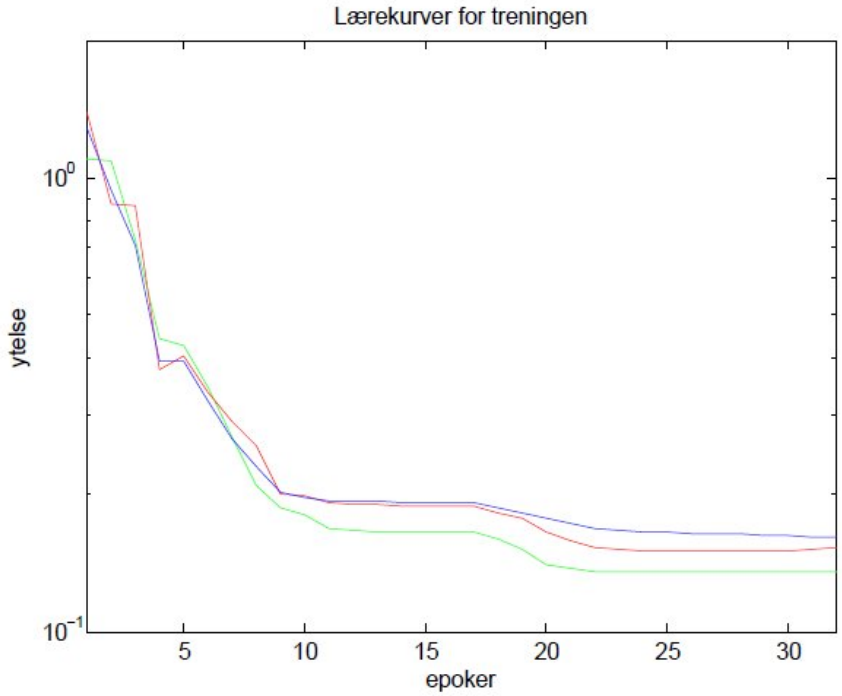
over et visst antall (her 20), blir ytelsen dårligere (se figur 4.4). Dette er fordi nettverket blir overtilpasset treningsdataene og vil da ikke fungere godt på nye data.



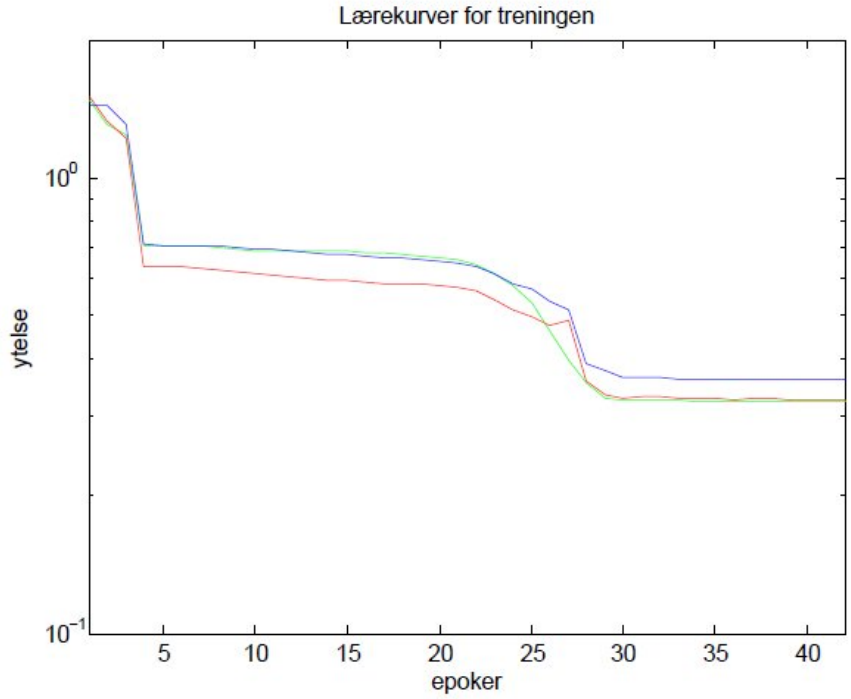
Figur 4.1: Lærekurver *2-3-2* nettverk.



Figur 4.2: Lærekurver *2-5-2* nettverk.



Figur 4.3: Lærekurver *2-20-2* nettverk.

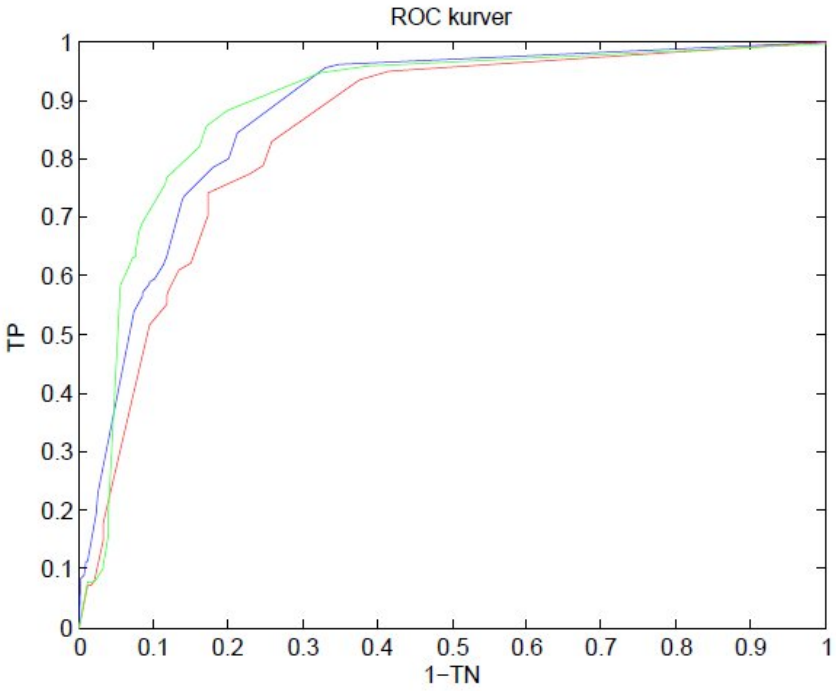


Figur 4.4: Lærekurver *2-30-2* nettverk.

Etter at terskelverdiene er funnet og TP, TN verdier er beregnet (se kapittel 3.6) kan en plotte ROC kurver som viser hvor godt de to egenskaper egner seg til å skille de mennesker/faste objekter fra støy. I figur 4.5 vises ROC kurver for *2-5-2* nettverket. Arealene under de ulike kurvene er beregnet. ATr betyr arealet under treningsdata (blå kurve), ATs betyr arealet under testdata (rød kurve) og Av betyr arealet under valideringsdata (grønn kurve). *2-5-2* nettverket har høy sannsynlighet for korrekt klassifikasjon samt størst areal under ROC kurvene. Dette tyder på at det er nettopp denne strukturen som vil egne seg best for å hjelpe SwimEye med å raffinere deteksjonen slik av den forbedres.

Et system som minimerer risikoen for å feilklassifisere mennesker/faste objekter må ha høy sensitivitet. Sensitivitet betegner andelen av sanne positive som blir korrekt klassifisert og spesifisitet betegner andel sanne negative som blir korrekt klassifisert. Hvis en for eksempel antar at en sensitivitet på 95 % er ønskelig, så kan en gå inn og lese av grafen i figur 4.5 at dette gir en sann negativ rate (spesifisitet) på 58,4 % (lest av for testsettet, rød kurve). Dette betyr at andel falske positive alarmer blir redusert med 58,4 %.

Resultatene tyder på at det er mulig å bruke nevrale nettverk til å bistå SwimEye med å raffinere deteksjonen av mennesker/faste objekter. De tester som har blitt gjort viser at et nevralt nettverk med fem skjulte noder er best egnet til å løse oppgaven.



Figur 4.5: ROC kurver *2-5-2* nettverk.
ATr = 0,880, Ats = 0,846, Av = 0,890.

5 Diskusjon

En god del av tiden er brukt til å hente ut analysebilder og implementere programmet, det ble da ikke tid til å undersøke om nettverksparametrenes verdier har noe å si på ytelsen for nettverket (alle eksperimenter er gjort med ”default” verdier). I ”Neural Network Toolbox” finnes en oversikt over blant annet ulike treningsfunksjoner, lærefunksjoner og transferfunksjoner, det anbefales derfor å se om nettverkets ytelse forbedres når disse funksjoner forandres.

Det kan stilles spørsmål om måten en finner terskelverdiene på kan være for dataavhengig. Det vil si at på grunn av for lite data vil terskelverdiene blir usikre og dermed blir også ROC analysen usikker. Kanskje burde en ha forsøkt å estimere de klassespesifikke sannsynlighetstetthetsfunksjonene eller også kalt aposteriori sannsynlighetene (se likning 5.1) i stede for å bruke nettverksutgangene som en direkte tilnærming disse.

$$p(x \mid \omega_i),\ i = 1,2 \quad (5.1).$$

Disse sannsynlighetene kan estimeres ved for eksempel maximum-likelihood [6], Parzen-vindu [6] eller nærmeste-nabo estimering [6].

Det kan også være aktuelt å implementere andre egenskapsbeskrivelser og teste ut om resultatet blir bedre med disse.

For å få enda bedre klassifisering kan det være aktuelt å forsøke å skille mennesker fra de faste objektene. Denne oppgaven har slått sammen mennesker og faste objekter til en klasse, å skille dem vil kunne føre til en mer presis klassifisering. For å kunne gjøre dette må en definere egenskaper som best mulig beskriver mennesker. Det trengs også flere analysebilder hvor det faktisk er et menneske som er på bildet.

En annen ting er at nettverkene i denne oppgaven analyserer ett og ett analysebilde om gangen, SwimEye analyserer en sekvens av bilder og sammenlikner med noen referansebilder. Disse referansebildene er bilder av den tomme bassengbunnen, altså helt sorte bilder uten noen hvite piksler. Det kan tenkes at det er mer effektivt å la nettverket jobbe på denne sekvensen av bilder. Støyen og sikten til kameraet kan variere fra basseng til basseng og i den forbindelse må det eventuelt trenes opp ett nettverk for hvert basseng slik at nettverket tilpasser seg forholdene i det gitte bassenget.

Det må nok forventes å få ulike resultater når en prøver det nevrale nettverket på helt nye data. Det er fordi det er jobbet med ganske lite data, hadde det vert tilgang på enda mer data ville en nok kunne forvente mindre avvik når en tester nettverket ut i felten.

6 Konklusjon

Resultatene i denne rapporten tyder på at data som er gitt for denne oppgaven vil kunne la seg klassifisere ved hjelp av et nevralt nettverk. Et nevralt nettverk med *2-5-2* struktur lar seg best bruke hvis en ønsker å sikre høy sann positiv rate. En får da en gjennomsnittlig sannsynlighet for korrekt klassifikasjon på 80,40 %. I følge ROC analysen vil et *2-5-2* nettverk gi en reduksjon i andel falske positive deteksjoner på 58,4 % ved 95 % sensitivitet. Det vil altså i 5 % av tilfellene gjøres en falsk negativ deteksjon. På grunnlag av dette vil et *2-5-2* nevralt nettverk anbefales å brukes til å gjøre SwimEyes deteksjon mer nøyaktig.

7 Referanseliste

1. Diverse skisse fra Davo AS.
Hjemmeside: http://www.davo.no/

2. Æsops fabel: Gutten som ropte ulv.

3. Tittel: Masteroppgave: Automatisk visuelt inspeksjonssystem.
Forfatter: Per Gunnar Bårdsen
Forlag: NTNU.
Tilgjengelig fra:
http://daim.idi.ntnu.no/masteroppgaver/IME/IDI/2006/1129/masteroppgave.pdf

4.
Tittel: NEURAL NETWORKS.
Forfatter: Christos Stergiou og Dimitrios Siganos
Tilgjengelig fra:
http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html#Why%20use%20neural%20networks

5. Tittel: Digital Image Processing.
Forfatter: Rafael C. Gonzalez og Richard E. Woods.
Forlag: Addison-Wesely Publishing Company.
ISBN: 0-201-50803-6.

6. Tittel: Pattern Classification Second Edition.
Forfatter: Richard O. Duda, Peter E. Hart og David G. Stork.
Forlag: JOHN WILEY & SONS, INC
ISBN: 0-471-05669-3.

7. Tittel: Modified Levenberg-Marquardt Method for Neural Networks Training.
Forfatter: Amir Abolfazl Suratgar, Mohammad Bagher Tavakoli og Abbas Hoseinabadi.
Forlag: WASET.ORG.
Tilgjengelig fra:
http://www.waset.org/pwaset/v6/v6-10.pdf.

8 Vedlegg

Som vedlegg gis en beskrivelse på hvordan de ulike funksjonene fungerer samt selve programkoden. Programkoden er skrevet i Matlab versjon R2007a.

8.1 Vedlegg A

Selve programkoden er skrevet i Matlab. I katalogen ”Neural Network Toolbox” har Matlab mange forhåndsdefinerte funksjoner som angår nevrale nettverk. Denne rapporten har brukt en del av disse funksjonene og de vil være merket med (M) etter navnet. I det etterfølgende kommer en forklaring på hvordan de enkelte funksjonene i programmet virker.

main3

Main er hovedfunksjonen, det er denne som kaller alle de andre funksjonene. Etter at *main* er kjørt har Matlab generert og trent opp et nevralt nettverk. Brukeren kan selv sette nettverksparametrene og treningsparametrene. *Main* vil vise lærekurvene for treningen, plott av ROC verdier og plott av egenskapsrommet.

lastInnData2

Denne funksjonen returnerer egenskapsvektoren P og målvektoren T. Først kalles funksjonen *hentBilder*, deretter genereres P og T. P inneholder verdier fra de ulike egenskapsuttrekningene og T inneholder de verdiene det nevrale nettverket skal trenes opp mot. I T er menneske definert som

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

og støy definert som

$$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

hentBilder

Funksjonen *hentBilder* laster inn bildene fra katalogen *analyseBilder5* og omgjør dem fra jpeg format til binære bilder, dette gjøres ved å kalle funksjonen *im2bw*(M). *HentBilder* returnerer to bildesett, obj1 som er binære bilder og obj2 som er de samme binære bildene bare at det har blitt utført morfologiske opperasjoner på dem.

egenskapC

egenskapC beregner gjennomsnittlig avstand mellom pikslene i bildet.

egenskapE

egenskapE beregner eksentrisiteten til bildet. Måten dette gjøres på er forklart i kapitel 2.1.1.

morf

Morf utfører en morfologisk åpning på bildesettet.

setNet

setNet kaller funksjonen *newff* (M) som genererer et nytt bakoverforplantnings nevralt nettverk.

dividevec (M)

Denne funksjonen deler datasettet inn i treningssett, valideringssett og et uavhengig testsett. Hvor stor andel av datasettet som skal gå til validering og testing bestemmes av parametrene *valProsent* og *testProsent*.

train (M)

Train trener opp den nevrale nettverket som er generert av *newff*.

sim (M)

Sim simulerer et nevralt nettverk.

beregnCM

Denne funksjonen beregner og returnerer klassifikasjonsmatrisen eller også kalt forvirringsmatrisen CM. CM inneholder følgende:

- *TP* antall sanne positive (menneske klassifisert som menneske).
- *FN* antall falske negative (menneske klassifisert som støy).
- *FP* antall falske positive (støy klassifisert som menneske).
- *TN* antall sanne negative (støy klassifisert som støy).

For å beregne CM så må en først finne ut hvor det største elementet i nettverksutgangene er. Til dette brukes matlabfunksjonen *max*.

Deretter bergenes differansen:

$$\max(\textit{målvektorene}) - 2 * \max(\textit{nettveksutgangene})$$

$\begin{bmatrix}1\\0\end{bmatrix} - 2\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}-1\\0\end{bmatrix}$ dette vil gi -1 i differansen og tilsvare TP.

$\begin{bmatrix}1\\0\end{bmatrix} - 2\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}1\\-2\end{bmatrix}$ dette vil gi -3 i differansen og tilsvare FN.

$\begin{bmatrix}0\\1\end{bmatrix} - 2\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\-1\end{bmatrix}$ dette vil gi -2 i differansen og tilsvare TN.

$\begin{bmatrix}0\\1\end{bmatrix} - 2\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}-2\\1\end{bmatrix}$ dette vil gi 0 i differansen og tilsvare FP.

CM er definert som:

$$\begin{bmatrix}TP & FN\\FP & TN\end{bmatrix}$$

vis

Denne funksjonen viser følgende resultater:

- Sjansen for korrekt klassifikasjon (P(korrekt deteksjon)).
- Sensitivitet, spesifitet og gjennomsnittet av disse.
- CM.
- Gjennomsnittlig antall epoker i treningen.
- Varians i P(korrekt klassifisering) for treningssett, tesesett og valideringssett.
- Gjennomsnittlig andel korrekt klassifisering i treningssett, tesesett og valideringssett.

egenskapsPlott

EgenskapsPlott viser et plott av egenskapsrommet for egenskapene beskrevet i kapittel 2.1.1.

getTh3

Beregner terskelverdiene, θ, i treningssettet.

Terskel8

Beregner TP og TN rater for treningssett, valideringssett og testsett.

Beregn

Beregner gjennomsnittlig *TP* og *TN* rate for de ulike settene.

ROC4

Viser ROC kurver for nettverk trent en gang.

meanROC

Viser gjennomsnittlige ROC verdier og arealet under kurvene.

visLarerkurver

Viser lærekurvene for trenings, validerings og tesesettet. Dette vises for nettverk trent en gang.

8.2 Vedlegg B

main3

```
function main3
%%%%%%%%%%hovedfunksonen genererer og simulerer etnevralt nettverk%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%%
clear all

[P,T]=lastInnData2;
[P,ps] = mapminmax(P);           %input data |P|,blir normalisert til
området -1 +1
egenskapsPlott(P);
%%%%%%%%%%%%%%%%%%%%%%%%%sett ulike nettverks parametere%%%%%%%%%%%%%%%%%%%%

S=[];
TF={};
antOutputs=length(T(:,1));        %antall outputs
d=length(P(:,1));                 %antall noder i inngangslag
n1=5;                             %antall noder i skjult lag 1
S3=antOutputs;                    %antall noder i utgangslag
TF1='tansig';                     %transferfunksjon for inngangslag
TF2='tansig';                     %transferfunksjon for skjult lag 1
TF3='tansig';                     %transferfunksjon for utgangslag
BTF='trainlm';                    %treningsfunksjon
BLF='learnwh';                    %læringsfunksjon

%%%%%%%%%%%%%%%%%%%%%%%%%sett ulike læreparametere%%%%%%%%%%%%%%%%%%%%%%%%

LP.lr=0.01;                        %lærerate, default 0.01
```

```
LP.mc=0.9;                          %moment konstant, default 0.9
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
S=[d n1 S3]; TF={TF1 TF2 TF3};
%%%%%%%%%%%%%%%%%%%%%%%%%%sett ulike treningsparametere%%%%%%%%%%%%%%%%%%%%%%

net.trainParam.epochs=100;          %max antall treningsepoker, default 100
net.trainParam.goal=0;              %treningsmålet, default 0
net.trainParam.max_fail=5;          %max antall valideringsfeil, default 5
net.trainParam.mem_reduc=1;         %"Factor to use for memory/speed
tradeoff", default 1
net.trainParam.min_grad=1e-10;      %minimum performance gradient, default
1e-10
net.trainParam.mu=0.001;            %initiell mu, default 0.001
net.trainParam.mu_dec=0.1;          %minskefaktor til mu, default 0.1
net.trainParam.mu_inc=10;           %økningsfktor til mu, default 10
net.trainParam.mu_max=100;          %max mu, default 100
net.trainParam.show=25;             %"Epochs between displays (NaN for no
displays)", default 25
net.trainParam.time=inf;            %treningstid i sekund, default inf
valProsent=0.3;                     %andel av P brukt til validering
testProsent=0.3;                    %andel av P brukt til testing
antallTreninger=10;                 %antall ganger nettverket trenes
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%siden det nevrale nettverket starter med tilfeldige verdier på vektene%%
%%%%vil resultatet variere for hver gang programmet kjører, for å unngå%%%%
%%%%%%%%%%denne tilfeldigheten i vektene settes følgende komando%%%%%%%%%%
rand('seed', 491218382)
%%%%%%%%%%genererer et " feedforward backpropagation " nettvek%%%%%%%%%%

net=setNet(P,S,TF,BTF,BLF);
%net=setNet2(P,T,S,TF,BTF,BLF);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%genererer treningssett (trS),kryssvalideringssett (cvS) og testsett(tstS)%
%%%%%%%%%% og trener nettverket med de genererte settene%%%%%%%%%%%%%%%%%%
TpTren1={};
TnTren1={};
TpTest1={};
TnTest1={};
TpVal1={};
TnVal1={};

for k=1:antallTreninger
[trS, cvS, tstS] = dividevec(P,T, valProsent,testProsent);
[net,tr] = train(net, trS.P, trS.T, [], [], cvS, tstS);
trSP=trS.P;
tstSP=tstS.P;
cvSP=cvS.P;

index=length(tr.epoch);
epochs(k)=tr.epoch(index);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%simulerer nettverk med de ulike datasettene%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%og viser resultatene%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[out,pf,af,E1,prf] = sim(net, tstS.P);       %%%uavhengig testsett
[out2,pf2,af2,E2,prf2] = sim(net, cvS.P);   %%%valideringssettet
[out3,pf3,af3,E3,prf3] = sim(net, trS.P);   %%%treningssettet
```

```
settTarget=tstS.T;
settTarget2=cvS.T;
settTarget3=trS.T;
[cm,N,y_out,I_out,y_t,I_t,diff]=beregnCM(out,settTarget);
[cm2,N2,y_out2,I_out2,y_t2,I_t2,diff2]=beregnCM(out2,settTarget2);
[cm3,N3,y_out3,I_out3,y_t3,I_t3,diff3]=beregnCM(out3,settTarget3);

trVperf=tr.vperf;
trTperf=tr.tperf;
trperf=tr.perf;


[korrekte]=vis(cm,N,trVperf,trTperf,trperf,[],[],[],[],[],[],k);
[korrekte2]=vis(cm2,N2,[],[],[],[],[],[],[],[],[],[],k);
[korrekte3]=vis(cm3,N3,[],[],[],[],[],[],[],[],[],[],k);


KorrekteTest(k)={korrekte};
KorrekteValidering(k)={korrekte2};
KorrekteTrening(k)={korrekte3};

PerfTest(k)={prf};
PerfValidering(k)={prf2};
PerfTrening(k)={prf3};
if k==1
    Th=getTh3(net,settTarget3,trSP);
end
[TPtren,TNtren,TPtest,TNtest,TPval,TNval]=terskel8(trSP,tstSP,cvSP,net,Th,s
ettTarget,settTarget2,settTarget3);
TpTren1(k)={TPtren};
TnTren1(k)={TNtren};
TpTest1(k)={TPtest};
TnTest1(k)={TNtest};
TpVal1(k)={TPval};
TnVal1(k)={TNval};

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
antEpoker=length(tr.epoch);
if antallTreninger==1
    visLarerkurver(trVperf,trTperf,trperf,antEpoker);
    ROC4(trSP,tstSP,cvSP,net,Th,settTarget,settTarget2,settTarget3);
    [korrekte]=vis(cm,N,trVperf,trTperf,trperf,[],[],[],[],[],[],k);
    [korrekte2]=vis(cm2,N2,[],[],[],[],[],[],[],[],[]);
    [korrekte3]=vis(cm3,N3,[],[],[],[],[],[],[],[],[]);

end
if antallTreninger>1

[meanTPtren,meanTntren,meanTPtest,meanTntest,meanTPval,meanTnval]=beregn(Tp
Tren1,TnTren1,TpTest1,TnTest1,TpVal1,TnVal1,antallTreninger);

[korrekte,varKorrekteTest,varKorrekteValidering,varKorrekteTrening,meanKorr
ekteTest,meanKorrekteValidering,meanKorrekteTrening]=vis(cm,N,trVperf,trTpe
rf,trperf,KorrekteTest,KorrekteValidering,KorrekteTrening,PerfTest,PerfVali
dering,PerfTrening,[],epochs);

meanROC(meanTPtren,meanTntren,meanTPtest,meanTntest,meanTPval,meanTnval);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

lastinnData2

```
function [P,T]=lastInnData2
%%%%%%%%%%laster inn dataene fra bildesettet og henter egenskapene%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%%%%

[obj1,obj2]=hentBilder;
%%%%%%%%%%%%%%%%%%%%%%%%%%%egenskaps vektor%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

P=[];
C=egenskapC(obj2);
E=egenskapE(obj2);
for i=1:numel(obj2)
    P(i,1)=C(i);
    P(i,2)=E(i);
end
P=P';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%målvektor%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

T=[];

T(1,1)=1;
T(1,2)=0;

T(2,1)=1;
T(2,2)=0;

T(3,1)=1;
T(3,2)=0;

T(4,1)=1;
T(4,2)=0;

T(5,1)=1;
T(5,2)=0;

T(6,1)=1;
T(6,2)=0;

T(7,1)=1;
T(7,2)=0;

T(8,1)=1;
T(8,2)=0;

T(9,1)=1;
T(9,2)=0;

T(10,1)=1;
T(10,2)=0;

T(11,1)=1;
T(11,2)=0;
```

```
T(12,1)=1;
T(12,2)=0;

T(13,1)=1;
T(13,2)=0;

T(14,1)=1;
T(14,2)=0;

T(15,1)=1;
T(15,2)=0;

T(16,1)=1;
T(16,2)=0;

T(17,1)=1;
T(17,2)=0;

T(18,1)=1;
T(18,2)=0;

T(19,1)=1;
T(19,2)=0;

T(20,1)=1;
T(20,2)=0;

T(21,1)=1;
T(21,2)=0;

T(22,1)=1;
T(22,2)=0;

T(23,1)=1;
T(23,2)=0;

T(24,1)=1;
T(24,2)=0;

T(25,1)=1;
T(25,2)=0;

T(26,1)=1;
T(26,2)=0;

T(27,1)=1;
T(27,2)=0;

T(28,1)=1;
T(28,2)=0;

T(29,1)=1;
T(29,2)=0;

T(30,1)=1;
T(30,2)=0;
```

```
T(31,1)=1;
T(31,2)=0;

T(32,1)=1;
T(32,2)=0;

T(33,1)=1;
T(33,2)=0;

T(34,1)=1;
T(34,2)=0;

T(35,1)=1;
T(35,2)=0;

T(36,1)=1;
T(36,2)=0;

T(37,1)=1;
T(37,2)=0;

T(38,1)=1;
T(38,2)=0;

T(39,1)=1;
T(39,2)=0;

T(40,1)=1;
T(40,2)=0;

T(41,1)=1;
T(41,2)=0;

T(42,1)=1;
T(42,2)=0;

T(43,1)=1;
T(43,2)=0;

T(44,1)=1;
T(44,2)=0;

T(45,1)=1;
T(45,2)=0;

T(46,1)=1;
T(46,2)=0;

T(47,1)=1;
T(47,2)=0;

T(48,1)=1;
T(48,2)=0;

T(49,1)=1;
T(49,2)=0;
```

```
T(50,1)=1;
T(50,2)=0;

T(51,1)=1;
T(51,2)=0;

T(52,1)=1;
T(52,2)=0;

T(53,1)=1;
T(53,2)=0;

T(54,1)=1;
T(54,2)=0;

T(55,1)=1;
T(55,2)=0;

T(56,1)=1;
T(56,2)=0;

T(57,1)=1;
T(57,2)=0;

T(58,1)=1;
T(58,2)=0;

T(59,1)=1;
T(59,2)=0;

T(60,1)=1;
T(60,2)=0;

T(61,1)=1;
T(61,2)=0;

T(62,1)=1;
T(62,2)=0;

T(63,1)=1;
T(63,2)=0;

T(64,1)=1;
T(64,2)=0;

T(65,1)=1;
T(65,2)=0;

T(66,1)=1;
T(66,2)=0;

T(67,1)=1;
T(67,2)=0;

T(68,1)=1;
T(68,2)=0;
```

```
T(69,1)=1;
T(69,2)=0;

T(70,1)=1;
T(70,2)=0;

T(71,1)=1;
T(71,2)=0;

T(72,1)=1;
T(72,2)=0;

T(73,1)=1;
T(73,2)=0;

T(74,1)=1;
T(74,2)=0;

T(75,1)=1;
T(75,2)=0;

T(76,1)=1;
T(76,2)=0;

T(77,1)=1;
T(77,2)=0;

T(78,1)=1;
T(78,2)=0;

T(79,1)=1;
T(79,2)=0;

T(80,1)=1;
T(80,2)=0;

T(81,1)=1;
T(81,2)=0;

T(82,1)=1;
T(82,2)=0;

T(83,1)=1;
T(83,2)=0;

T(84,1)=1;
T(84,2)=0;

T(85,1)=1;
T(85,2)=0;

T(86,1)=1;
T(86,2)=0;

T(87,1)=1;
T(87,2)=0;
```

```
T(88,1)=1;
T(88,2)=0;

T(89,1)=1;
T(89,2)=0;

T(90,1)=1;
T(90,2)=0;

T(91,1)=1;
T(91,2)=0;

T(92,1)=1;
T(92,2)=0;

T(93,1)=1;
T(93,2)=0;

T(94,1)=1;
T(94,2)=0;

T(95,1)=1;
T(95,2)=0;

T(96,1)=1;
T(96,2)=0;

T(97,1)=1;
T(97,2)=0;

T(98,1)=1;
T(98,2)=0;

T(99,1)=1;
T(99,2)=0;

T(100,1)=1;
T(100,2)=0;

T(101,1)=1;
T(101,2)=0;

T(102,1)=1;
T(102,2)=0;

T(103,1)=1;
T(103,2)=0;

T(104,1)=1;
T(104,2)=0;

T(105,1)=1;
T(105,2)=0;

T(106,1)=1;
T(106,2)=0;
```

```
T(107,1)=1;
T(107,2)=0;

T(108,1)=1;
T(108,2)=0;

T(109,1)=1;
T(109,2)=0;

T(110,1)=1;
T(110,2)=0;

T(111,1)=1;
T(111,2)=0;

T(112,1)=1;
T(112,2)=0;

T(113,1)=1;
T(113,2)=0;

T(114,1)=1;
T(114,2)=0;

T(115,1)=1;
T(115,2)=0;

T(116,1)=1;
T(116,2)=0;

T(117,1)=1;
T(117,2)=0;

T(118,1)=1;
T(118,2)=0;

T(119,1)=1;
T(119,2)=0;

T(120,1)=1;
T(120,2)=0;

T(121,1)=1;
T(121,2)=0;

T(122,1)=1;
T(122,2)=0;

T(123,1)=1;
T(123,2)=0;

T(124,1)=1;
T(124,2)=0;

T(125,1)=1;
T(125,2)=0;
```

```
T(126,1)=1;
T(126,2)=0;

T(127,1)=0;
T(127,2)=1;

T(128,1)=0;
T(128,2)=1;

T(129,1)=0;
T(129,2)=1;

T(130,1)=0;
T(130,2)=1;

T(131,1)=0;
T(131,2)=1;

T(132,1)=0;
T(132,2)=1;

T(133,1)=0;
T(133,2)=1;

T(134,1)=0;
T(134,2)=1;

T(135,1)=0;
T(135,2)=1;

T(136,1)=0;
T(136,2)=1;

T(137,1)=0;
T(137,2)=1;

T(138,1)=0;
T(138,2)=1;

T(139,1)=0;
T(139,2)=1;

T(140,1)=0;
T(140,2)=1;

T(141,1)=0;
T(141,2)=1;

T(142,1)=0;
T(142,2)=1;

T(143,1)=0;
T(143,2)=1;

T(144,1)=0;
T(144,2)=1;
```

```
T(145,1)=0;
T(145,2)=1;

T(146,1)=0;
T(146,2)=1;

T(147,1)=0;
T(147,2)=1;

T(148,1)=0;
T(148,2)=1;

T(149,1)=0;
T(149,2)=1;

T(150,1)=0;
T(150,2)=1;

T(151,1)=0;
T(151,2)=1;

T(152,1)=0;
T(152,2)=1;

T(153,1)=0;
T(153,2)=1;

T(154,1)=0;
T(154,2)=1;

T(155,1)=0;
T(155,2)=1;

T(156,1)=0;
T(156,2)=1;

T(157,1)=0;
T(157,2)=1;

T(158,1)=0;
T(158,2)=1;

T(159,1)=0;
T(159,2)=1;

T(160,1)=0;
T(160,2)=1;

T(161,1)=0;
T(161,2)=1;

T(162,1)=0;
T(162,2)=1;

T(163,1)=0;
T(163,2)=1;
```

```
T(164,1)=0;
T(164,2)=1;

T(165,1)=0;
T(165,2)=1;

T(166,1)=0;
T(166,2)=1;

T(167,1)=0;
T(167,2)=1;

T(168,1)=0;
T(168,2)=1;

T(169,1)=0;
T(169,2)=1;

T(170,1)=0;
T(170,2)=1;

T(171,1)=0;
T(171,2)=1;

T(172,1)=0;
T(172,2)=1;

T(173,1)=0;
T(173,2)=1;

T(174,1)=0;
T(174,2)=1;

T(175,1)=0;
T(175,2)=1;

T(176,1)=0;
T(176,2)=1;

T(177,1)=0;
T(177,2)=1;

T(178,1)=0;
T(178,2)=1;

T(179,1)=0;
T(179,2)=1;

T(180,1)=0;
T(180,2)=1;

T(181,1)=0;
T(181,2)=1;

T(182,1)=0;
T(182,2)=1;
```

```
T(183,1)=0;
T(183,2)=1;

T(184,1)=0;
T(184,2)=1;

T(185,1)=0;
T(185,2)=1;

T(186,1)=0;
T(186,2)=1;

T(187,1)=0;
T(187,2)=1;

T(188,1)=0;
T(188,2)=1;

T(189,1)=0;
T(189,2)=1;

T(190,1)=0;
T(190,2)=1;

T(191,1)=0;
T(191,2)=1;

T(192,1)=0;
T(192,2)=1;

T(193,1)=0;
T(193,2)=1;

T(194,1)=0;
T(194,2)=1;

T(195,1)=0;
T(195,2)=1;

T(196,1)=0;
T(196,2)=1;

T(197,1)=0;
T(197,2)=1;

T(198,1)=0;
T(198,2)=1;

T(199,1)=0;
T(199,2)=1;

T(200,1)=0;
T(200,2)=1;

T(201,1)=0;
T(201,2)=1;
```

```
T(202,1)=0;
T(202,2)=1;

T(203,1)=0;
T(203,2)=1;

T(204,1)=0;
T(204,2)=1;

T(205,1)=0;
T(205,2)=1;

T(206,1)=0;
T(206,2)=1;

T(207,1)=0;
T(207,2)=1;

T(208,1)=0;
T(208,2)=1;

T(209,1)=0;
T(209,2)=1;

T(210,1)=0;
T(210,2)=1;

T(211,1)=0;
T(211,2)=1;

T(212,1)=0;
T(212,2)=1;

T(213,1)=0;
T(213,2)=1;

T(214,1)=0;
T(214,2)=1;

T(215,1)=0;
T(215,2)=1;

T(216,1)=0;
T(216,2)=1;

T(217,1)=0;
T(217,2)=1;

T(218,1)=0;
T(218,2)=1;

T(219,1)=0;
T(219,2)=1;

T(220,1)=0;
T(220,2)=1;
```

```
T(221,1)=0;
T(221,2)=1;

T(222,1)=0;
T(222,2)=1;

T(223,1)=0;
T(223,2)=1;

T(224,1)=0;
T(224,2)=1;

T(225,1)=0;
T(225,2)=1;

T(226,1)=0;
T(226,2)=1;

T=T';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

hentBilder

```
function [obj1,obj2]=hentBilder
%%%%%%%%%%%%laster inn bilder fra mappen analyseBilder%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%%%

bilder = dir(fullfile('F:','Master','analyseBilder5','*.jpg'));
d=zeros(numel(bilder),1);
obj1=cell(size(d));
obj2=cell(size(d));

for k = 1:numel(bilder)
    obj1(k) = {im2bw(imread(bilder(k).name))};
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
se1 = strel('disk',3,4);     %størrelse på åpne strukturelementet
obj2=morf(obj1,0,se1,[]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

morf

```
function [obj]=morf(obj,se1)
%%%%%%%%%%%%%%%%%%%%%%%%utfører morfologisk åpning%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:numel(obj)
%%%%%%%%%%%%%%%%%%%%%%%%%%morfologisk åpning%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    obj(i)={imopen(obj{i,1},se1)};

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Egenskap C

```
function [C]=egenskapC(obj2)
%%%%%%%%%%%finner gj. snittlig avstand mellom pikslene i bildet%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%

C=[];
d=zeros(numel(obj2),1);
b=cell(size(d));
d1=cell(size(d));
index1=cell(size(d));
nonZero1=cell(size(d));
for i=1:numel(obj2)
    if(isempty(find(double(obj2{i,1}))))
        d1(i)={zeros(1,114960)};
    else
        d1(i,1)={pdist(cell2mat(obj2(i,1)))};
    end
    index1(i,1)={find(cell2mat(d1(i,1)))};
    d2=cell2mat(d1(i,1));
    index2=cell2mat(index1(i,1));
    nonZero1(i,1)={d2(index2)};
end
C=cellfun(@mean, nonZero1); %finner gj.snittlig euklidisk avstand mellom
pikslene

%%%%%%%%%%%%%%%%%%%%erstatter eventuelle NaN i C med 0%%%%%%%%%%%%%%%%%%%%
for j=1:numel(C)
    if isnan(C(j))
        C(j)=0;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

egenskapE

```
function [E]=egenskapE(obj2)
%%%%%%%%%%%%%%%%%%%%beregner eksentrisitet til objektet%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%

for i=1:numel(obj2)

    if(isempty(find(double(obj2{i,1}))))
        stats1=0;
        stats2=1;

    else
    STATS1 = regionprops(double(obj2{i,1}),'MajorAxisLength');
    STATS2 = regionprops(double(obj2{i,1}),'MinorAxisLength');
    stats1=STATS1.MajorAxisLength;
    stats2=STATS2.MinorAxisLength;
    end
    E(i)=stats1/stats2;
end
E=E';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

egenskapsPlott

```
function egenskapsPlott(P)
```

```
%%%%%%%%%plotter egenskapsrommet, mennesker er markert med blå kryss%%%%%%%
%%%%%%%%%%%%%%%%%%%%og støy er markert med røde sirkler%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%%%

group=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2];
gscatter(P(1,:),P(2,:),group,'br','xo');
title('Plott av egenskapsrommet');
xlabel('Egenskap C');
ylabel('Egenskap E');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

setNet

```
function [net]=setNet(P,S,TF,BTF,BLF)
%%%%%%%%%%%%%%%%%%%%%%%%genererer et nevralt nettverk%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%%%

net = newff(minmax(P),S,TF,BTF,BLF);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

beregnCM

```
function [cm,N,y_out,I_out,y_t,I_t,diff]=beregnCM(out,settTarget)
%%%%%%%%%%%%%%%%%%%%%%%%%%beregner cm matrisen%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%%%

[y_out,I_out] = max(out);
[y_t,I_t] = max(settTarget);
diff = (I_t - 2*I_out);

TN = length(find(diff==-2));     % støy klassifisert som støy
FN = length(find(diff==-3));     % menneske klassifisert som støy
TP = length(find(diff==-1));     % menneske klassifisert som menneske
FP = length(find(diff==0));      % støy klassifisert som menneske
N = size(settTarget,2);          % antall test sampler
cm = [TP FN; FP TN];             % klassifikasjonsmatrise
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

vis

```
function
[korrekte,varKorrekteTest,varKorrekteValidering,varKorrekteTrening,meanKorr
ekteTest,meanKorrekteValidering,meanKorrekteTrening]=vis(cm,N,trVperf,trTpe
rf,trperf,KorrekteTest,KorrekteValidering,KorrekteTrening,PerfTest,PerfVali
dering,PerfTrening,k,epochs)
%%%%%%%%%%%%%%%%%%%%%%%%%viser resultater%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%%%

if nargin==11
  fprintf('Totalt antall testsampler: %d\n', N);
  fprintf('forvirringsmatrise');
```

```
  cm
  cm_p = (cm ./ N) .* 100 ;         % klassifikasjonsmatrise i prosent
  disp(['P(korrekt klassifikasjon) : ',num2str((cm(1,1)+cm(2,2))/N)]);
  disp(['Antall falske negative klassifikasjoner (FN) :
',num2str(cm(1,2))]);
end

korrekte=(cm(1,1)+cm(2,2))/N;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%beregner varians,stdavvik og gj.snitt for korrekt klassifikasjon%%%%%%
%%%%%%%%%%%%%%beregner gj. snittlig sensitivitet og spesifitet%%%%%%%%%%%%%%
%%%beregner gj.snittlig ytelse(performance) og gj.snittlig ant. epoker%%%%%
if nargin==13
    varKorrekteTest=var(cell2mat(KorrekteTest));
    varKorrekteValidering=var(cell2mat(KorrekteValidering));
    varKorrekteTrening=var(cell2mat(KorrekteTrening));


    meanKorrekteTest=mean(cell2mat(KorrekteTest));
    meanKorrekteValidering=mean(cell2mat(KorrekteValidering));
    meanKorrekteTrening=mean(cell2mat(KorrekteTrening));

    meanPerfTest=mean(cell2mat(PerfTest));
    meanPerfValidering=mean(cell2mat(PerfValidering));
    meanPerfTrening=mean(cell2mat(PerfTrening));

    meanEpochs=mean(epochs);

    disp(' ');
    disp(['varians i korekt klassifisering testsett :
',num2str(varKorrekteTest)]);
    disp(['gj.snittlig andel korrekt klassifisering testsett i % :
',num2str(meanKorrekteTest)]);
    disp(['varians i korekt klassifisering valideringssett :
',num2str(varKorrekteValidering)]);
    disp(['gj.snittlig andel korrekt klassifisering valideringssett i % :
',num2str(meanKorrekteValidering)]);
    disp(['varians i korekt klassifisering treningssett :
',num2str(varKorrekteTest)]);
    disp(['gj.snittlig andel korrekt klassifisering treningssett i % :
',num2str(meanKorrekteTrening)]);
    disp(['gj.snittlig antall epoker : ',num2str(meanEpochs)]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

getTh3

```
function [Th]=getTh3(net,settTarget3,trSP)
%%%%%%%%%%%%%%%%%%%%%%%%%beregner terskelverdier%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%

ggT=sim(net,trSP);
GGT=ggT(1,1:length(ggT))./ggT(2,1:length(ggT));
[GT,indxT] = sort(GGT);
GT=GT';
GGT=GGT';
indxT=indxT';
```

```
%%%%%%%%%%%%%%%%%%%%finner hvor i P elementene i trSP er%%%%%%%%%%%%%%%%%%%%
indGGT=zeros(length(settTarget3),1);
for i=1:length(indGGT)
     a1=settTarget3(1,i);
     a2=settTarget3(2,i);
     if a1>a2
        indGGT(i)=1;
     end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%finner klassefasit for sorterte data%%%%%%%%%%%%%%%%%
indxGT=zeros(length(GT),1);
grenseTren=length(find(indGGT));
for i=1:length(indxT)
    if indxT(i)<=grenseTren
       indxGT(i)=1;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%finner terskelverdier%%%%%%%%%%%%%%%%%%%%%%%%%%
Thtab=zeros(length(indxGT),1);
for i=1:length(indxGT)-1
    if isequal(indxGT(i),indxGT(i+1))==0
        Thtab(i)=i;
    end
Thtab2=find(Thtab);
Th=zeros(length(Thtab2)+2,1);
Th(1)=GT(1)-4000;
Th(length(Thtab2)+2)=GT(length(GT))+4000;
for i=1:length(Thtab2)
    Th(i+1)=GT(Thtab2(i))+((GT(Thtab2(i)+1)-GT(Thtab2(i)))/2);
end
end
Th=flipud(Th);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

terskel8

```
function
[TPtren,TNtren,TPtest,TNtest,TPval,TNval]=terskel8(trSP,tstSP,cvSP,net,Th,s
ettTarget,settTarget2,settTarget3)
%%%%%%%%%%%%%%%%%%%%Finner TN og TP rater for de ulike settene%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%%%

ggT=sim(net,trSP);
GGT=ggT(1,1:length(ggT))./ggT(2,1:length(ggT));
[GT,indxT] = sort(GGT);
GT=GT';
GGT=GGT';
indxT=indxT';
%%%%%%%%%%%%%%%%%%%%finner hvor i P elementene i trSP er%%%%%%%%%%%%%%%%%%%%
indGGT=zeros(length(settTarget3),1);
for i=1:length(indGGT)
     a1=settTarget3(1,i);
     a2=settTarget3(2,i);
     if a1>a2
        indGGT(i)=1;
     end
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%finner klassefasit for sorterte data%%%%%%%%%%%%%%%%%
indxGT=zeros(length(GT),1);
grenseTren=length(find(indGGT));
for i=1:length(indxT)
    if indxT(i)<=grenseTren
       indxGT(i)=1;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
totW1tren=length(find(indxGT));
totW2tren=length(indxGT)-totW1tren;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GT=flipud(GT);
indxGT=flipud(indxGT);
%%%%%%%%%%%%%%%%%%%%%%%%finner TP og TN rate i treningssett%%%%%%%%%%%%%%%%%
tptren=[];
tntren=[];
for i=1:length(Th)
    indxTPtren=find(GT>Th(i));
    indxTNtren=find(GT<Th(i));
    tptren(i)=length(find(indxGT(indxTPtren)));
    tntren(i)=length(find(indxGT(indxTNtren)==0));
end
TPtren=tptren./totW1tren;
TNtren=tntren./totW2tren;
TPtren=fliplr(TPtren);
TNtren=fliplr(TNtren);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ggTest=sim(net,tstSP);
GGTest=ggTest(1,1:length(ggTest))./ggTest(2,1:length(ggTest));
[GTest,indxTest] = sort(GGTest);
GTest=GTest';
GGTest=GGTest';
indxTest=indxTest';
%%%%%%%%%%%%%%%%%%%%%%%%finner hvor i P elementene i trSP er%%%%%%%%%%%%%%%%%%
indGGTest=zeros(length(settTarget),1);
for i=1:length(indGGTest)
     a1=settTarget(1,i);
     a2=settTarget(2,i);
     if a1>a2
        indGGTest(i)=1;
     end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%finner klassefasit for sorterte data%%%%%%%%%%%%%%%%%
indxGTest=zeros(length(GTest),1);
grenseTest=length(find(indGGTest));
for i=1:length(indxTest)
    if indxTest(i)<=grenseTest
       indxGTest(i)=1;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
totW1test=length(find(indxGTest));
totW2test=length(indxGTest)-totW1test;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GTest=flipud(GTest);
indxGTest=flipud(indxGTest);
%%%%%%%%%%%%%%%%%%%%%%%%finner TP og TN rate i testsettet%%%%%%%%%%%%%%%%%%%%
```

```
tptest=[];
tntest=[];
for i=1:length(Th)
    indxTPtest=find(GTest>Th(i));
    indxTNtest=find(GTest<Th(i));
    tptest(i)=length(find(indxGTest(indxTPtest)));
    tntest(i)=length(find(indxGTest(indxTNtest)==0));
end
TPtest=tptest./totW1test;
TNtest=tntest./totW2test;
TPtest=fliplr(TPtest);
TNtest=fliplr(TNtest);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ggTval=sim(net,cvSP);
GGTval=ggTval(1,1:length(ggTval))./ggTval(2,1:length(ggTval));
[GTval,indxTval] = sort(GGTval);
GTval=GTval';
GGTval=GGTval';
indxTval=indxTval';
%%%%%%%%%%%%%%%%%%%%finner hvor i P elementene i trSP er%%%%%%%%%%%%%%%%%%%%
indGGTval=zeros(length(settTarget2),1);
for i=1:length(indGGTval)
     a1=settTarget2(1,i);
     a2=settTarget2(2,i);
     if a1>a2
        indGGTval(i)=1;
     end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%finner klassefasit for sorterte data%%%%%%%%%%%%%%%%%%%%
indxGTval=zeros(length(GTval),1);
grenseTval=length(find(indGGTval));
for i=1:length(indxTval)
    if indxTval(i)<=grenseTval
       indxGTval(i)=1;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
totW1val=length(find(indxGTval));
totW2val=length(indxGTval)-totW1val;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
GTval=flipud(GTval);
indxGTval=flipud(indxGTval);
%%%%%%%%%%%%%%%%%%%%finner TP og TN rate i valideringssettet%%%%%%%%%%%%%%%%%%
tpval=[];
tnval=[];
for i=1:length(Th)
    indxTPval=find(GTval>Th(i));
    indxTNval=find(GTval<Th(i));
    tpval(i)=length(find(indxGTval(indxTPval)));
    tnval(i)=length(find(indxGTval(indxTNval)==0));
end
TPval=tpval./totW1val;
TNval=tnval./totW2val;
TPval=fliplr(TPval);
TNval=fliplr(TNval);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

visLarerkurver

```
function visLarerkurver(trVperf,trTperf,trperf,antEpoker)
%%%%%%%%%%%%%%%%%%%%%%viser lærekurver for de ulike settene%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%%%

   figure;
   semilogy(trVperf,'g');
   hold on;
   semilogy(trTperf,'r');
   semilogy(trperf,'b');
   xlabel('epoker');
   ylabel('ytelse');
   title('Lærekurver for treningen')

   xmin=1;
   xmax=antEpoker;
   ymin=10^-1;
   ymax=2;
   axis([xmin xmax ymin, ymax])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

ROC4

```
function ROC4(trSP,tstSP,cvSP,net,Th,settTarget,settTarget2,settTarget3)
%%%%%%%%%%%%%%%%%%%%%%plotter ROC kurver for en trening%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%%%

[TPtren,TNtren,TPtest,TNtest,TPval,TNval]=terskel8(trSP,tstSP,cvSP,net,Th,s
ettTarget,settTarget2,settTarget3);
figure;
plot(1.-TNtren,TPtren,'b');
xlabel('1-TN');
ylabel('TP');
hold on
plot(1.-TNtest,TPtest,'r');
plot(1.-TNval,TPval,'g');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

beregn

```
function
[meanTPtren,meanTntren,meanTPtest,meanTntest,meanTPval,meanTnval]=beregn(Tp
Tren1,TnTren1,TpTest1,TnTest1,TpVal1,TnVal1,antallTreninger)
%%%%%%%%%%%%%%%%%%%%%%%%Finner gj.snittlige ROC verdier%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%%%

END=length(TpTren1{1});
TpTrenMat=zeros(antallTreninger,END);
TnTrenMat=zeros(antallTreninger,END);
TpTestMat=zeros(antallTreninger,END);
TnTestMat=zeros(antallTreninger,END);
TpValMat=zeros(antallTreninger,END);
TnValMat=zeros(antallTreninger,END);
for j=1:antallTreninger
   TpTrenMat(j,:)=cell2mat(TpTren1(1,j));
   TnTrenMat(j,:)=cell2mat(TnTren1(1,j));
   TpTestMat(j,:)=cell2mat(TpTest1(1,j));
   TnTestMat(j,:)=cell2mat(TnTest1(1,j));
   TpValMat(j,:)=cell2mat(TpVal1(1,j));
```

```
   TnValMat(j,:)=cell2mat(TnVal1(1,j));
end
meanTPtren=mean(TpTrenMat);
meanTntren=mean(TnTrenMat);
meanTPtest=mean(TpTestMat);
meanTntest=mean(TnTestMat);
meanTPval=mean(TpValMat);
meanTnval=mean(TnValMat);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

meanROC

```
function
meanROC(meanTPtren,meanTntren,meanTPtest,meanTntest,meanTPval,meanTnval)
%%%%%%%%%%%%%%%%%%%%%%%%plotter gj.snittlige ROC verdier%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%Gaute Dirdal Lunde 28.05.2009%%%%%%%%%%%%%%%%%%%%%%

figure;
plot(1.-meanTntren,meanTPtren,'b');
xlabel('1-TN');
ylabel('TP');
title('ROC kurver')
hold on
plot(1.-meanTntest,meanTPtest,'r');
plot(1.-meanTnval,meanTPval,'g');
hold off
disp(['Areal under ROC kurve for testsett : ',num2str(-trapz(1.-
meanTntest,meanTPtest))])
%disp(['Areal under ROC kurve for treningssett : ',num2str(-trapz(1.-
meanTntren,meanTPtren))]);
%disp(['Areal under ROC kurve for valieringssett : ',num2str(-trapz(1.-
meanTnval,meanTPval))]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```