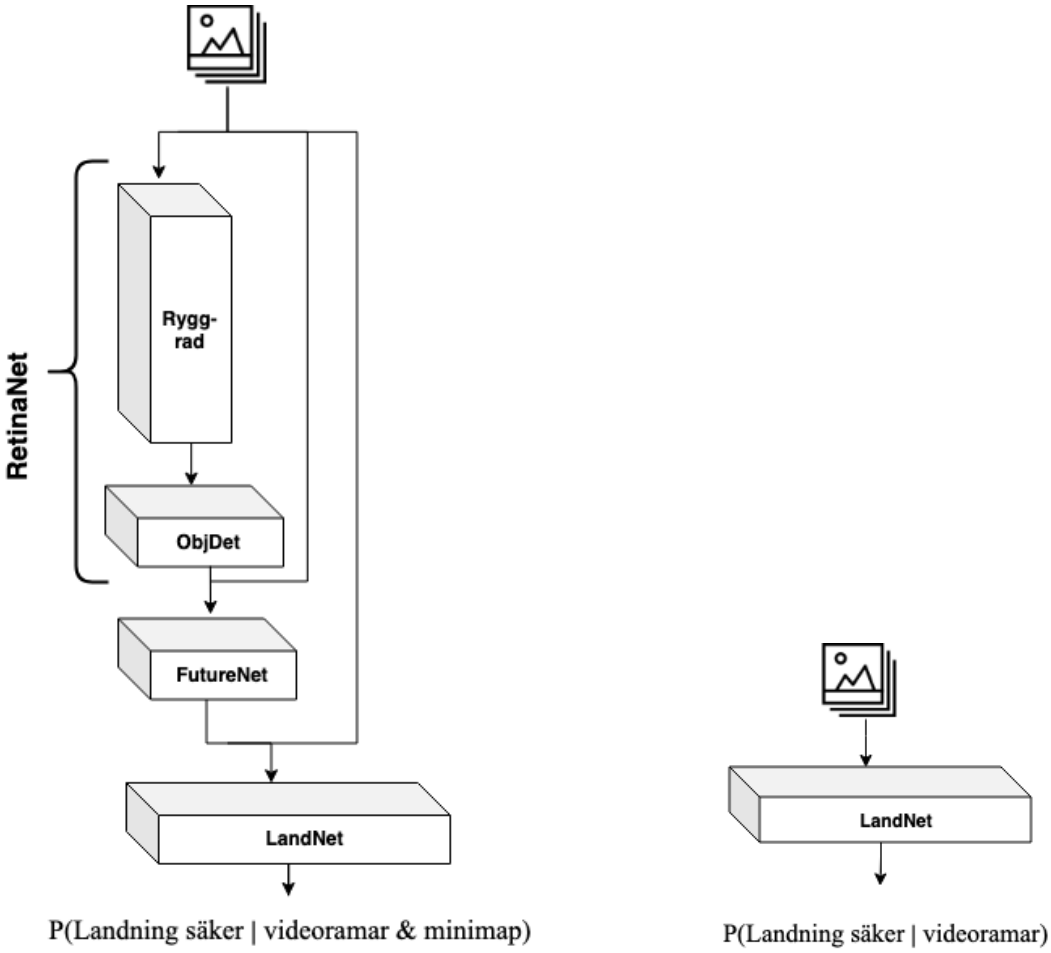
CHALMERS



En säker landning av drönare via ett artificiellt neuralt nätverk

Examensarbete inom Data- och informationsteknik

Adam Hyttring
Hannes Gustafsson

Institutionen för Data- och informationsteknik
Chalmers tekniska högskola
Göteborgs universitet
Göteborg, Sverige 2019

Examensarbete 2019

En säker landning av drönare via ett artificiellt neuralt nätverk

Utforskar möjligheten att förbättra intelligensen i drönare för att automatiskt bedöma om det är säkert att utföra en landning.

Adam Hyttring
Hannes Gustafsson

Institutionen för Data- och informationsteknik
Chalmers tekniska högskola
Göteborgs universitet
Göteborg, Sweden 2019

En säker landning av drönare via ett artificiellt neuralt nätverk
Utforskar möjligheten att förbättra intelligensen i drönare för att automatiskt bedöma om det är säkert att utföra en landning.
Adam Hyttring
Hannes Gustafsson

Handledare:
Jonas Duregård, Institutionen för Data- och informationsteknik
Albin Falk, Sigma Technology Development

Examinator: Peter Lundin, Institutionen för Data- och informationsteknik

Institutionen för Data- och informationsteknik
Chalmers tekniska högskola/ Göteborgs Universitet
SE-412 96 Göteborg
Sverige
Telefon: +46 (0)31-772 1000

Omslag:
Illustration av två olika neurala nätverks-modeller som utvärderas i arbetet.

Typsatt i LaTeX
Göteborg, Sweden 2019

A safe landing of drones via a artificial neural network
Exploring the capability of extending the intelligence of drones for automatically assess the safty of landing.

Adam Hyttring & Hannes Gustafsson
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

Modern drones are often supplied with a built-in landing function. The function is usually primitive and rarely involves any advanced risk assessment. In this report, we propose a solution to transfer the human component of the risk assessment to the drone in the form of a neural network.

It has previously been found advantageous to make valuable information more explicit before using it as an input in a neural network. Two different types of network models are compared in the work. One network only uses the drone's camera frames as input variables, reducing its complexity. The second network performs more transformations on the drone's camera frames to first extract valuable information from the images to make them explicit. The network then uses the explicit information in combination with the drone's frames as input variables. In this report, a broader comparison is also presented about what impact on the network's precision more explicit data has.

The result of the work shows that networks that use explicit, valuable information have higher precision, but that the magnitude of the precision difference depends on the properties of the task. We also show that a neural network can, with good precision, perform a risk assessment of a drone's landing.

Keywords: Artificial Intelligence, Automatic landing, Neural network, Explicit/Implicit information, Drones

Sammanfattning

Moderna drönare levereras ofta med en inbyggd landningsfunktion. Funktionen är ofta primitiv och involverar sällan någon avancerad riskbedömning. I denna rapport föreslår vi en lösning för att överföra den mänskliga komponenten i riskbedömningen till drönaren i form av ett neuralt nätverk.

Det har, i tidigare arbeten, visat sig vara fördelaktigt att göra värdefull information mer explicit vid användning av ett neuralt nätverk. I arbetet jämförs två olika sorters nätverksmodeller. Ett nätverk tar endast in drönarens kamera-ramar som inmatnings-variabler, vilket minskar dess komplexitet. Det andra nätverket utför fler transformationer av drönares kamera-ramar för att först extrahera värdefull information ur bilderna för att göra dem explicita. Nätverket använder sedan den explicita informationen i kombination med drönar-ramar som inmatnings-variabler. I rapporten presenteras även en bredare jämförelse kring vilken påverkan på nätverkets precision mer explicit data har.

Resultatet av arbetet påvisar att ett neuralt nätverk kan, med god precision, utföra en riskbedömning av en drönares landning. Vi visar även att nätverk som använder expliciterad värdefull information har högre precision men att storleken på precisions-skillnad beror på uppgiftens egenskaper

Nyckelord: Artificiell Intelligens, Automatisk landning, Neurala nätverk, Explicit/Implicit information, Drönare

Förord

Rapporten behandlar kandidatarbetet Landning via ett Artificiellt Neuralt Nätverk av drönare, som genomfördes på Chalmers tekniska högskola under vårterminen 2019.

Vi vill tacka vår handledare Albin Falk och Sigma Technology Development för den frihet som gavs vid val av arbete och allt stöd. Vi vill även ge ett stort tack till Jonas Duregård, utan honom skulle rapporten vara strukturlös, och allmänt mindre intressant att läsa.

Ordlista

Explicit information I arbetet används begreppet explicit information som konkretisering av nödvändig information i en bild.

Implicit information Implicit information används som motsats-begrepp till explicit information. Modellen måste lära sig att internt extrahera nödvändig information ur en bild.

Mitt-till-Mitt I arbetet används Mitt-till-Mitt-inlärning istället för det engelska ordet "Mid-to-Mid learning"

Bakåtpropagering I rapporten översätts ordet "Backpropagation" till bakåtpropagering.

Framåtmatande I rapporten översätts ordet "Feed forward" till framåtmatande.

Uppdatering I rapporten används ordet uppdatering för att indikera på att en justering av alla nätverkets vikter har utförts.

Helt konvolutionellt neuralt nätverk I rapporten används begreppet "Helt konvolutionellt neuralt nätverk" som översättning för samlingsnamnet "Fully convolutional neural network"

Ryggrad I rapporten används begreppet "ryggrad" genomgående. I detta arbete avser "ryggrad" det helt konvolutionella neurala nätverk som extraherar bild-egenskaper som sedan används av projektets objekt-detektor.

Innehåll

1 Inledning **1**

2 Underliggande teori **4**

3 Metod **11**

3.1 Verktyg för mjukvaruutveckling 11

3.2 Implementation av drönare 14

3.3 Datamängder 16

3.4 Modelarkitektur 18

3.5 Experiment 21

3.6 Träning av modeller 23

4 Resultat & Diskussion **27**

5 Slutsats **32**

5.1 Möjligt fortsatt arbete 32

Referenser **37**

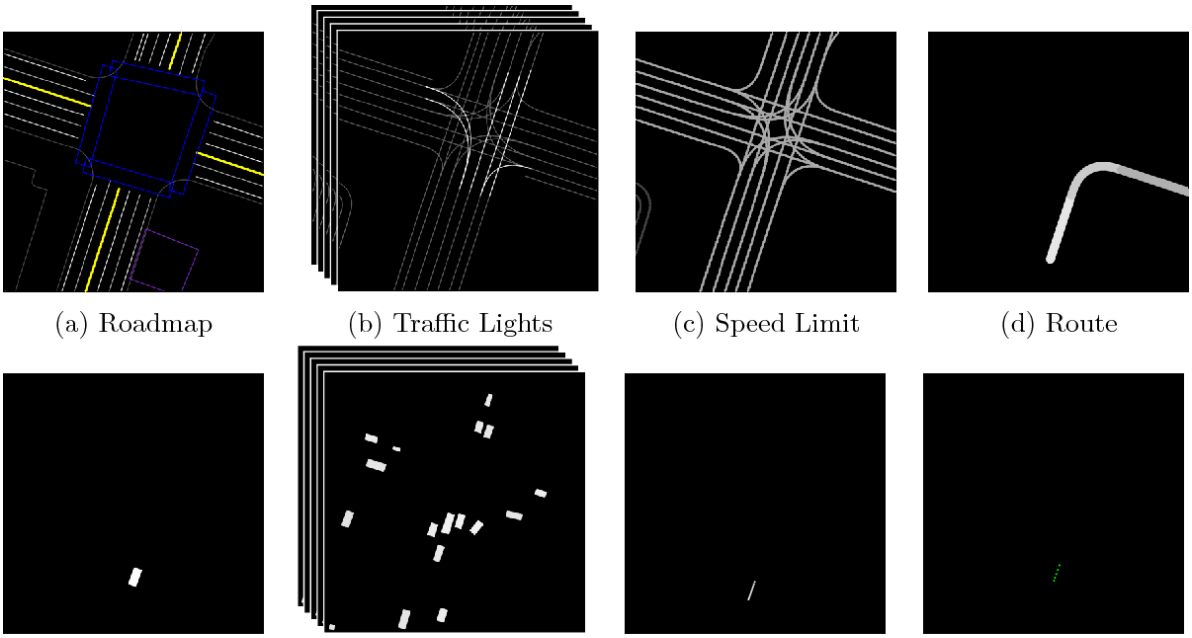
1

Inledning

Projektet kommer att fokusera på utvecklingsområden inom dagens drönar-teknik. Moderna drönare levereras ofta med en inbyggd landningsfunktion. Denna funktion är ofta primitiv och involverar sällan någon avancerad riskbedömning. Detta leder till att mänsklig översikt krävs vid landning. Problemet som skall lösas i detta arbete är att göra det säkert att automatiskt landa en drönare. Säkert för drönaren, människor i dess närhet samt diverse egendom. Problemet skall lösas via en implementation av ett djupt neuralt nätverk.

För att undersöka om neurala nätverk är en gångbar lösning för riskbedömningar av drönar-landningar undersöks två modeller av olika arkitekturer. Den ena modellen använder början-till-slut-principer medan den andra använder uppgifts-specifika-nätverk för att först extrahera värdefull information. Idén för modellen som nyttjar uppgifts-specialiserade nätverket är till stor del inspirerat av Waymos ChauffeurNet [7].

ChauffeurNet använder redan behandlar data som inmatningsvariabler istället för icke-behandlad data, så som foton från kameror. Fördelen med denna sorts arkitektur är bland annat möjligheten att nyttja uppgifts-specialiserade nätverk med syfte att omvandla svårstrukturerad data till explicit värdefull information. Exempel på sådant kan ses i figur 1. I (a) kan en förenklad bild ses av en vägkorsning, producerat av ett neuralt nätverk. Att bilden förenklas innebär att information som inte medför fördelaktig kunskap skalas bort. Denna bortskalning av överflödig information innebär att en effektiviserar nätverket genom att endast låta det extrapolera på värdefull information så som den bild-information som presenterades i figur 1 (a) [28].



(a) Roadmap (b) Traffic Lights (c) Speed Limit (d) Route

(e) Current Agent Box (f) Dynamic Boxes (g) Past Agent Poses (h) Future Agent Poses

Figur 1: Illustration över de inmatningsvariabler som Waymos ChauffeurNet använder [7].

Syfte

Syftet med projektet är att undersöka hur en kan överföra ansvaret för riskbedömningen, om huruvida en drönare kan säkert landa, från människan till drönaren genom ett artificiellt neuralt nätverk. Tekniken kan bidra till kommersiella applikationer av drönarteknik, så som hemleveranser av fysiska produkter.

Avgränsningar

Drönaren som kommer att användas under projektet skall köpas in från bolaget DJI. Detta för att avgränsa projektet till endast mjukvaruutveckling och att DJI förser en god plattform för utveckling mot deras drönare. DJI förser projektet med drönarens styrteknik och ingen ytterligare utveckling av tekniken förväntas.

Det neurala nätverket kommer att utvecklas med hjälp av Googles verktyg Tensorflow. Nätverket kommer köras på en dators grafikkort och kommer därmed inte köras i drönaren.

Ytterligare avgränsningar som är värda att nämna är exempelvis att vägar kommer ses som en säker plats att landa på när det inte finns något objekt på vägen, ett problem som skulle kunna lösas med en GPS (Global Positioning System).

Frågeställningar

Projektet fokuserar på att besvara fyra frågeställningar, dessa är:

1. Med hur god precision kan ett djupt neuralt nätverk förutspå huruvida en drönare kan utföra en säker landning?

Neurala nätverk har i tidigare arbeten visat sig väl lämpade på att lösa binära klassificeringsuppgifter [31]. Kan en nyttja denna teknik för att överföra ansvaret för riskbedömningen från människa till drönare?

2. Uppstår det några skillnader vid träning och utvärdering av ett neuralt nätverk genom att göra värdefull information mer explicit?

Det har i vissa fall visat sig vara gynnsamt att göra värdefull information mer explicit för neurala nätverk [28]. Kan en finna liknande resultat i detta arbete?

3. Kan en genom ett helt konvolutionellt nätverk och positions-subtraktion förutspå en persons framtida position?

Det har visat sig vara möjligt att, med hjälp av ett återkopplat konvolutionellt neuralt nätverk, förutspå bilars framtida position [32]. I detta arbete undersöks hur väl ett, endast, konvolutionellt neuralt nätverk kan förutspå var en persons framtida position kommer att vara.

4. Kan en applicera överföring av lärande på objektdetektering på en vitt skild domän, som drönare i detta arbete, och är vissa modeller mer lämpade för en sådan sorts uppgift än andra?

Överföring av lärande har blivit populärt inom dator-syn [4]. Datamängder som modeller tränas på tenderar att innehålla få bilder tagna från hög höjd vinklade neråt, därmed är drönar-domänen vitt skild från de redan tränade modellerna. Då drönare har denna sorts synvinkel kan det uppstå problem med att återanvända och fin-justera redan tränade modeller.

2

Underliggande teori

I kapitlet presenteras den bakgrundsinformation och teorier som ligger till grund för projektet.

Artificiell Intelligens

Artificiell Intelligens (AI) är när en intelligent agent är uppbyggd så att den själv kan göra handlingar som främjar sitt mål. Det finns idag flera olika typer av AI, som kan delas in i två större kategorier, klassisk AI som bygger på regelbaserade resonemangssystem och statistisk AI där mönster och trender hittas via statistiska verktyg som ofta är kopplade med maskininlärning.

Maskininlärning

Maskininlärning bygger på att få datorer att lära sig hur den skall agera utifrån erfarenhet av data. Maskininlärning är ett samlingsnamn för de algoritmer datorer använder för att hitta mönster i data de exponeras för.

Övervakad maskininlärning Målet med övervakad maskininlärning är att producera en funktion som parar ihop inmatningsvariabler med utmatningsvariabler på ett önskat sätt. Det kallas för övervakad då majoriteten av träningen som utförs på modellen görs med inmatningsvariabler som har annoterats med önskade utmatningsvariabler [1].

Oövervakad maskininlärning Oövervakad maskininlärning är när en har kända inmatningsvariabler utan några korresponderade utmatningsvariabler. Målet med oövervakad maskininlärning är att identifiera en underliggande struktur i inmatningsvariablerna för att kunna extrahera användbar information ur dem [1].

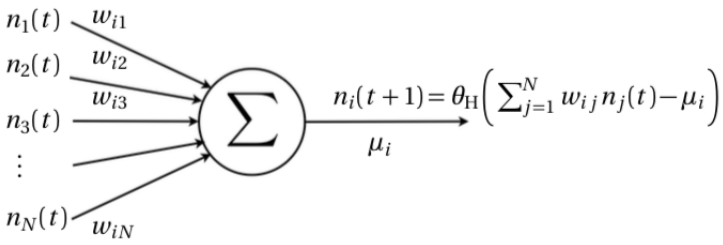
Början-till-slut-inlärning I många maskininlärnings-modeller delas behandlingen av data upp i olika steg och deluppgifter. För varje modell som utför en deluppgift sker en form av extrahering av data-egenskaper och sedan en, eventuell behandling av datan. Den behandlade datan används sedan som inmatningsvariabler till nästa modell [2]. Början-till-slut-modeller avstår helt från denna sorts uppdelning. All behandling av data sker i en enda modell. Modellen tar in givna inmatningsvariabler och producerar en slutlig förutsägelse [2].

Mitt-till-mitt-inlärning Mitt-till-mitt-inlärning (MtM) är en variation av början-till-slut-inlärning. I MtM används redan behandlad data som inmatningsvariabler till modellen och den producerar förutsägelser som förväntas behandlas av en annan modell. Behandling av datan kan till exempel vara en heurstisk funktion så som väg-planering.[7].

Neurala nätverk

Neurala nätverk är ett samlingsnamn för en grupp av maskininlärnings-algoritmer. Bearbetningssystemet är inspirerat av hur det biologiska nervsystemet behandlar data.

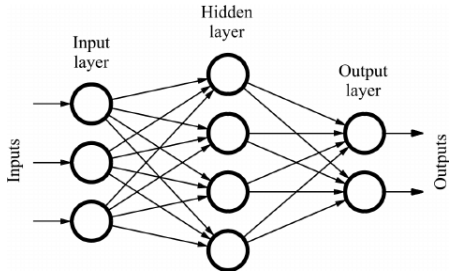
Neurala nätverk är uppbyggda av flera, mindre behandlings-enheter, så kallade neuroner. Neuronerna är inspirerade av så kallade McCulloch Pitts-neuroner [3]. Varje neuron tar emot en eller flera inmatningsvariabler. Variablerna påverkar neuroner olika mycket beroende på hur stark kopplingen är mellan variabelns källa och neuronen. Källan är i många fall också en neuron. Styrkan på kopplingen mellan två neuroner kallas för vikt och är en variabel som kan förändras genom att nätverket tränas. Neuroner behandlar inmatningsvariablerna och producerar en enda utmatningsvariabel [3]. I figur 2 illustreras hur neuronerna behandlar inmatningsvariabler för att producera en utmatningsvariabel. Neuronerna kopplas sedan ihop för att bilda ett nätverk genom att neuroner gruppers i form av olika lager, se figur 3. Hur inmatningsvariablerna och den behandlade datan propagerar genom nätverket beror på vilken sorts nätverket en har.



Figur 2: En schemantisk illustration av en neuron. Låt oss kalla den stora neuronen för huvudneuronen, annoterat som i i figuren. n_x annoterar en neuron som är kopplad till huvudneuronen. w_{ix} är styrkan av kopplingen mellan huvudneuronen och neuron n_x där $w_{ix} \in \mathbb{R}$. Huvudneuronen summerar alla ingående variabler multiplicerat med styrkan av kopplingen mellan huvudneuronen och inmatningsvariablens källa. Därefter appliceras en aktiveringsfunktion på summan, annoterat som θ_H i figuren. Resultatet genererar huvudneuronens utmatningsvariabel. Aktiveringsfunktion beskrivs senare i rapporten. t annoterar i vilken turordning neuronerna utför beräkningar [3].

Urval av problem som kan lösas med neurala nätverk Det finns många olika sorters problem som kan lösas med hjälp av neurala nätverk. Två vanliga problem är klassificerings-problem och regressions-problem. Klassificeringsproblem är problem som involverar att förutspå vilken klass-annotation en eller flera givna inmatningsvariabler tillhör. Det kan till exempel vara att avgöra om en bild illustrerar en katt eller hund, eller om det är säker att utföra en landning. Regressionsproblem handlar om att nätverket skall förutspå en kontinuerlig kvantitet med avsende på en eller flera givna variabler. Ett exempel på ett sådant problem är att förutspå priset på en bostad utifrån dess egenskaper.

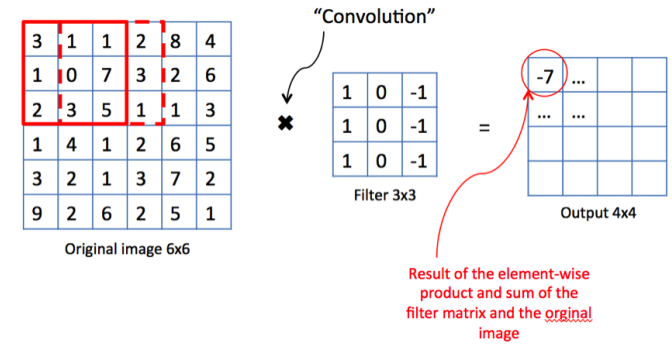
Framåtmatande neurala nätverk Ett framåtmatande neurala nätverk är ett nätverk där kopplingarna mellan neuronerna inte producerar en cykel. Varje neuron tar emot det föregående lagrets neuroners utmatningsvariabler som ingångsvariabel. Denna sorts framåtmatande kallas även för framåtpropagerande. Figur 3 illustrerar hur neuronerna propagerar varaiblerna framåt.



Figur 3: Uppbyggnaden av ett framåtmatande neuralt nätverk [3].

Konvolutionellt neuralt nätverk Konvolutionellt neuralt nätverk utvecklades under 1980-talet. Denna sort av nätverk blev kända och brett använda efter att de använts för att vinna ImageNet-tävlingen 2012 (ImageNet är en tävling inom datorsyn som har använts som riktmärke för att jämföra olika tekniker inom området).

Konvolutionella neurala nätverk är specifikt designade objektdetektering och mönsterigenkänning. Designen har viktiga egenskaper inspirerade från hur syn processeras i en mänsklig hjärna. Den första egenskapen är en rumslig matris som dras över bilden. Matrisen är uppbyggd av samma sorts neuroner som i ett vanligt framåtmatande neuralt nätverk. Matrisen kallas även för filter. Filterna är ofta rektangelformade, till exempel 3x3. Dessa filter appliceras på olika delar av bilden [16]. I Figur 4 kan en illustration hittas som visar hur filtret appliceras på en bild. Den andra egenskapen är att neuronernas vikter justeras för att upptäcka bilders lokala egenskaper så som kanter eller mönster [3].



Figur 4: Illustration av hur ett filter appliceras på en bild [16]. Filtret innehåller 3x3 neuroner som har samma egenskaper som McCulloch-Pitts-neuroner. De fyllda injerna i orginalbilden visar hur stor andel av bilden som filtret appliceras på. De streckade linjerna visar hur filtret kommer att förflyttas över bilden och sedan upprepa bearbetningsprocessen [16].

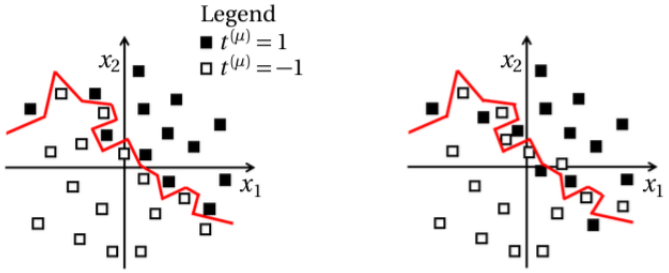
Träning Som tidigare nämnts kallas styrkan mellan neuroner för vikter. När det sägs att ett nätverk tränas innebär det att en iterativ sökning utförs för att identifiera styrkorna mellan neuroner sådan att nätverket uppvisar önskat betende. Önskat betende kan till exempel vara att klassificera olika bilder som äpplen eller päron [12]. Sökningen innebär att vikterna mellan neuronerna justeras, oftast en väldigt liten justering. En utförd justering kallas för uppdatering i resten av rapporten.

Bakåtpropagering Bakåtpropagering är tekniken som används för att beräkna derivatorna på det neurala nätverkets förlustfunktion med avsende på dess vikter. Det görs genom att iterativt applicera kedjeregeln, som är en regel för att beräkna derivatan av sammansatta funktioner vilket ett neurala nätverk i grund och botten är. Derivatan används för att, genom en given teknik, optimera vikternas styrka med mål att minimera förlustfunktionen. Tekniken är viktig då den ligger till grund för hur majoriteten av olika sorters neurala nätverk lär sig [9].

Aktiveringsfunktion Inom neurala nätverk är aktiveringsfunktionen den funktion som definierar en neurons utgångsvariabel givet dess ingångsvariabler. I figur 2 är denna funktion annoterad som θ_H [6]. Valet av aktiveringsfunktion baseras på vilken sorts uppgift nätverket skall utföra. I arbetet skall, till exempel, ett nätverk räkna ut sannolikheten för en säker landning, och använder därmed aktiveringsfunktionen Sigmoid som ger värden mellan noll och ett.

Förlustfunktion Förlustfunktionen beräknar hur stor avvikelsen är mellan vad det neurala nätverket förutspådde och det förväntade värdet. Det neurala nätverkets vikter justeras med mål att minimera värdet av förlustfunktionen [8]. Kan generellt beskrivas som en utvärderingsfunktion.

Överanpassning Vid träning av ett neuralt nätverk är målet att nätverket skall anpassas utifrån de egenskaper som datamängden påvisar. Ett problem som kan uppstå är överanpassning. Överanpassning innebär att nätverkets vikter anpassas efter egenskaper som datamängden nätverket tränas på innehar men som inte nödvändigtvis existerar utanför mängden [3]. Målet med ett nätverk bör inte vara att prestandan ska maximeras på den datamängd den tränas på. Målet bör vara att nätverket påvisar att den generaliserar väl och kan prestera bra på datamängder den ej exponerats för. En orsak till att överanpassning kan ske är att nätverket har för många parametrar [3]. Tidigare arbeten har däremot visat att fler parametrar generellt leder till högre topp-precision [31]. En kan se ett exempel på hur överanpassning kan se ut nedan i figur 5.



Figur 5: Illustration av hur överanpassning kan se ut på ett två-dimensionellt problem. Till vänster syns hur väl en icke-linjär gräns lyckas separera data, ska påvisa att ett nätverk som överanpassats presterar väl på data som ingår bland träningsmängden. Till höger syns hur små förändring på datapunkterna får nätverket att missklasificera [3].

Återkallelse och precision I rapporten används begreppen återkallelse, Recall, och precision återkommande. De är två vanliga metoder som används vid utvärdering av modeller med binära utmatningsvariabler. Återkallelse är andelen av alla, annoterat, sanna variabler som identifierats som sanna. Precision är den andel av identifierat sanna variabler som även är annoterade som sanna. Högre precision och återkallelse är önskvärt. Ekvationerna presenteras nedanför. Om en modell ska klassificera variabler med 1or och 0or gäller följande. TP: Korrekt 1a, FP: Felaktig 1a, TN: Korrekt 0a, FN: Felaktig 0a.

$$Precision = \frac{TP}{TP + FP} \quad \text{(Precision)}$$

$$Recall = \frac{TP}{TP + FN} \quad \text{(Återkallelse)}$$

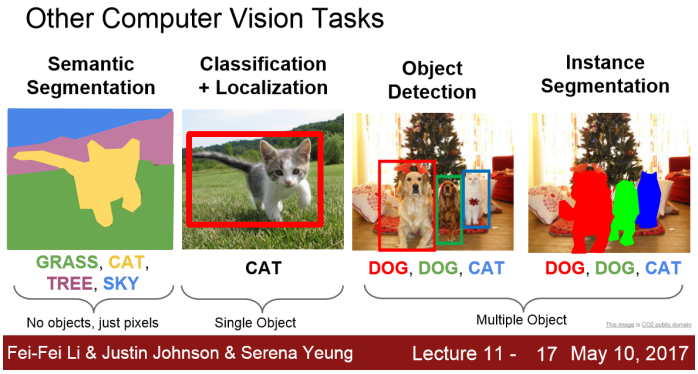
Överföring av lärande Överföring av lärande innebär att en modell, som originellt är utvecklad för att lösa en unik uppgift, återanvänds för att lösa en ny fast liknande uppgift. Att låta en modell återanvändas kan innebära att en ny modell kan uppnå god prestanda fortare och i vissa fall bättre topp-prestanda än om den hade tränats från grunden [4]. Till exempel kan det göras genom att kopiera vikterna på en modell som har tränats på att klassificera en väldigt stor mängd bilder och objekt. Vikterna ska sedan endast finjustera genom träning på andra bilder och objekt som det originella nätverket inte tränats på.

Djup maskininlärning Som tidigare beskrivit är artificiella neurala nätverk uppbyggda av olika lager. Om ett nätverk har mer än tre lager totalt klassificeras det som ett djupt nätverk [3]. Varje lager innebär att inmatningsvariabeln genomgår en icke-linjär transformation med målet att i slutändan kunna representeras på en abstrakt nivå och ett meningsfullt vis [5].

Objektdetektering Objektdetektering är en datorseende-teknik med fokus på att upptäcka ett eller flera objekt i en bild som tillhör en mängd av utvalda klasser. En objektdetekteringsalgoritm kan tillexempel vara specifikt framtagen för att detektera kaffekoppar [11]. Figur 6 illustrerar skillnaden mellan olika tekniker som på ett eller annat sätt fokuserar på att fastställa objekts närvaro i bilder.

Objektlokalisering Objektlokalisering är en datorseende-teknik med mål på att producera koordinaterna för var det intressanta objektet är i bilden. Den andra bilden från höger i figur 6 illustrerar tekniken. Den stora skillnaden mellan detektering och lokalisering är att lokalisering gäller strikt ett objekt medans detektering kan hantera flera [11].

Bildklassificering Bildklassificering är en datorseende-teknik med mål att avgöra vad för diskret klass en bild innehåller. Ett exempel på en sådan uppgift är att låta en modell avgöra om en bild innehåller en katt eller hund.



Figur 6: Illustration av olika datorseende-tekniker som beskrivits ovan[34].

Annotering av data

För att ett övervakat nätverk skall tränas på insamlad data skapas annotationer till den datan. Annotationerna tillsammans med den ursprungliga datan bör innehålla tillräcklig information så nätverket har möjlighet att lära sig. Det finns flera olika sätt att annotera data på, ett exempel är annotationer om positioner i en bild där olika objekt befinner sig

Avgränsande rektanglar Ett sätt att annotera bilder är att rita rektanglar runt objekt som önskas ingå bland de klasser som nätverket bör upptäcka. En rektangel innehar informationen om dess position i en bild och vilken objekttyp som rektangeln formas runt, till exempel om det är en person eller om det är en bil. Exempel på hur dessa rektanglar kan se ut kan ses i Figur 6 andra och tredje bilden från vänster.

Semantisk segmentering Vid semantisk segmentering annoteras varje pixel i en bild till förutbestämda objektklasser. Då varje pixel är annoterad, finns information om hela bilden. Exempelvis finns de exakta positionerna i en bild som anses tillhöra objektklasserna “Grass”,“Cat”,“Tree” och “Sky” som visas i figur 6 första bilden från vänster.

3

Metod

I kapitlet, metod, beskrivs hur projektet kan återproduceras.

3.1 Verktyg för mjukvaruutveckling

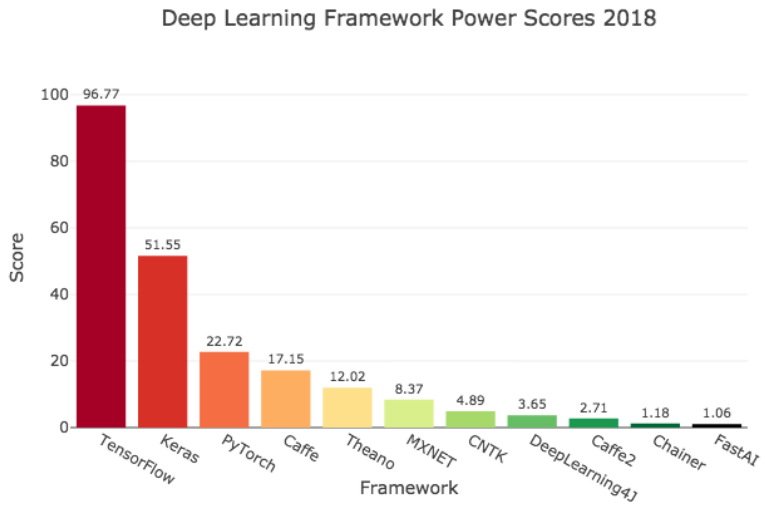
I kapitlet presenteras den hårdvara och de mjukvaru-resurser som användes i projektet.

Bibliotek för utveckling av neurala nätverk

I projektet användes ett antal verktyg för att underlätta utvecklingen av neurala nätverk. Verktygen möjliggjorde snabbare framtagning av prototyper av nätverk och tillförde därmed effektivare fel-diagnosering och optimering.

Tensorflow Tensorflow är en plattform skapad för utveckling och implementering av maskininlärnings-modeller. Plattformen har en öppen källkod. Användningen av plattformen har vuxit under de senaste åren och är idag världen mest använda för utveckling av neurala nätverk [23]. Se figur 7 för en jämförelse gällande användningen av olika ramverk för neurala nätverk.

Keras Keras är ett högnivå-API som producerar en mer abstrakt syntax vid utveckling av neurala nätverk jämfört med till exempel Tensorflow. Keras är för sig inte en maskininlärnings-plattform utan är byggt att köras ovanpå plattformar så som Tensorflow [24]. Huvuddelen av utvecklingen av det neurala nätverket i detta arbete skrevs i Python och med hjälp av Keras.



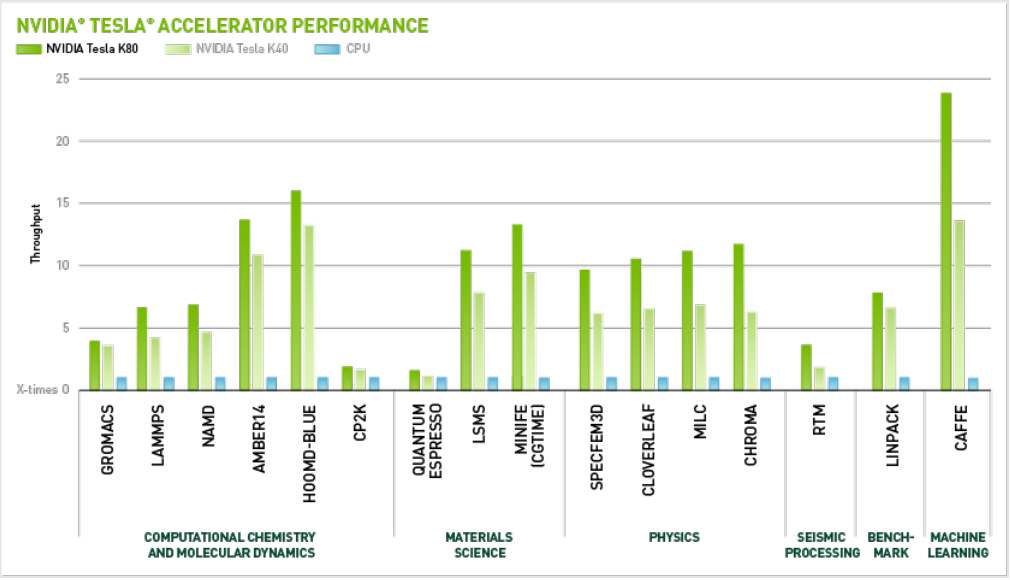
Figur 7: Grafen visar hur användningen av olika plattformar för utveckling av neurala nätverk såg ut under 2018. Poängen är beräknade utifrån en rad olika egenskaper, så som antal google-sökning, GitHub-aktivitet och hur vanlig plattformen var vid publicerade vetenskapliga artiklar [25].

Parallella operationer exekverade på grafikkort

Neurala nätverk är i princip seriellt utförda matris-operationer där matriser är inmatningsvariabler och vikter. Många matris-operationer är väl lämpade att utföras parallellt. Operationer som kan utföras parallellt kräver många mindre kraftfulla processorer istället för färre men mer kraftfulla. Detta innebär att grafikkort är mer lämpade för denna sorts arbete än vanliga processorer då grafikkort har många fler kärnor (mindre processorer) [9]. Utveckling för grafikkretsar kan skilja sig markant från utveckling för vanliga processorer. För att undvika att skriva egna lågnivå-funktioner som drar nytta av grafikkorts stora förmåga av parallell-databehandling används olika bibliotek och plattformar. I arbetet användes grafikkortet GTX1060 för att behandla den data som samlades in av drönaren.

Cuda Cuda är en plattform för parallell databehandling och programmering framtagen av Nvidia. Plattformen tillåter utvecklare att dra nytta av grafikkorts fördelar vid parallella operationer. Jämfört med processorer har grafikkort fler kärnor. Fler kärnor innebär att operationer som tillåts att utföras parallellt kan, i många fall, utföras snabbare [18]. I Figur 8 visas hur processor och grafikkorts prestanda skiljer sig stort inom funktioner som utförs parallellt.

cuDNN cuDNN är ett bibliotek framtaget av Nvidia specifikt för utveckling av neurala nätverk och andra djupmaskininlärnings-tekniker. Biblioteket innehåller optimerade implementationer av standard-funktioner så som matrismultiplikationer. cuDNN drar nytta av Cuda-plattformen för att möjligöra, och optimera, parallella databehandling. Skillnaderna mellan Cuda och cuDNN är att Cuda är ett bibliotek för att kommunicera med grafikkort och cuDNN är ett bibliotek med optimerade funktioner framtaget för utveckling av neurala närverk [17].



Figur 8: I grafen visas hur processor presterar i jämförelse med grafikkort på vanliga operationer som utförs parallellt. Jämförelsen använder en processor som index. Längst till höger kan en se att maskininlärnings-biblioteket Caffe presterar nästan 25x högre än en vanlig processor. Grafikkorten är utvecklade av Nvidia. [35]

3.2 Implementation av drönare

I det här kapitlet beskrivs de implementationer som behövdes för att kunna samla in data med en drönare, där datan var av tillräcklig hög kvalité så ett neuralt nätverk effektivt kan lära sig utav den.

Drönare

I projektet krävs en drönare för att samla in data och testa nätverket. Drönaren som valts är DJI's minsta drönare som har en vikt på 80 gram inkluderar egenskaper så som WIFI 802.11n 2.4G, 720P Live View och barometer [13]. Drönaren är även utrustad med två antenner för att göra kommunikationen extra stabil för att kunna skicka en videoström och samtidigt kunna ta emot kommandon.

Kamera Drönaren Tello är utrustad med en HD720P30 kamera vilket ger en tillräckligt bra bildkvalité för att samla in data som ett neuralt nätverk kan tränas på. Kameran på drönaren är dock riktad rakt framåt vilket inte är optimalt för projektet då drönaren behöver kunna se neranför sig.

Infraröd-sensor Tello är även utrustad med en IR-sensor som möjliggör en mjukare landning då den känner av när det närmar sig en yta neranför den [14].

Modifiering av drönare

Som tidigare skrivet valdes drönaren Tello av företagen DJI och Ryze. De två stora nackdelarna med detta val var kamerans riktning som hade ett synfält rakt framåt samt begränsningen av hur brett synfältet var. För att lösa dessa problem monterades kameran ur sin position så den kunde se rakt ner och ett fisköga monterades ovanpå kameran. Figur (a) nedan visar hur drönaren såg ut innan modifieringar, figur (b) nedan visar drönaren efter modifieringar.

(a) Orginalversionen av drönaren.

(b) Modifierad version av drönaren.

Kommunikation mellan drönare och dator

Tello som har wifi-stöd kan ta emot och skicka information via UDP-paket (User Datagram Protocol). En länk skapas mellan drönare, som agerar server och en dator , som agerar klient. Tello's API talar om vilka kommandon som Tello kan ta emot och behandla.

Codec videoram

Drönaren skickar videoramar i form av en byteström som är öppen mellan drönaren och datorn. Strömmen av bytes behöver avkodas till videoramar och detta kan göras med video codec.

3.3 Datamängder

För att träna neurala nätverk krävs en stor mängd annoterad data [9]. Tillgång till denna data kan nås via olika tillvägagångsätt. I projektet diskuterades tre olika sätt att samla in data på, simulerad data, offentlig data och egeninsamlad data.

Simulerad data

Genom en simulerad miljö kan en erhålla en stor mängd data. Metoden verkar lovande då insamling av en större mängd egeninsamlad data är tidskrävande. I projektet valdes metoden att inte användas på grund av de svårigheter som finns att försöka ersätta en verklig miljö med en simulerad [30].

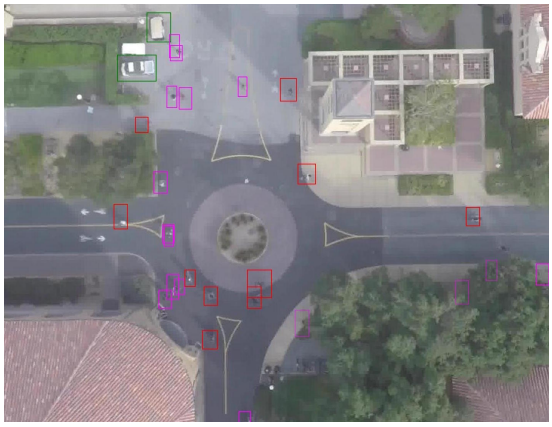
Offentlig data

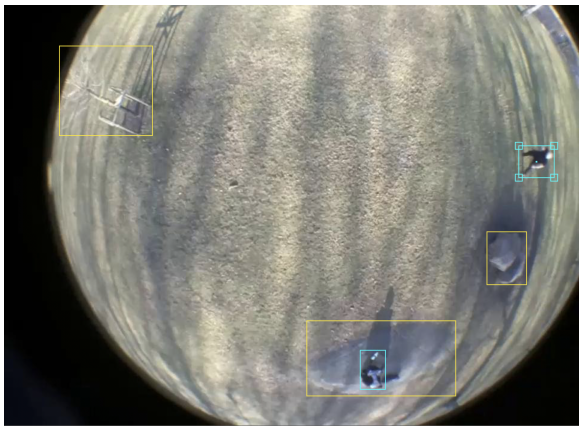
En datamängd som övervägdes att användas var ett som var framtaget av Stanford [15]. Datamängden innehåller redan annoterad data av typen avgränsande rektanglar och klassificering av rektanglarna. Ett beslut togs däremot att inte använda datamängden med anledningar som berör olikheter mellan kamerabilderna tagna av drönaren som användes i projektet och bilderna i Stanfords dataset. En godtycklig bild tagen från datamängden kan ses i Figur 10 tillsammans med en bild tagen från drönaren.

Egeninsamlad data

Metoden som valdes var att samla in egen data för att erhålla en träningsmängd så lik den av drönarens bilder. Insamlingen av data skedde iterativt under projektets gång. Före varje datainsamlings-tillfälle diskuterades var bilderna borde tas och vilka objekt som ansågs vara viktiga att känna igen. Tillvägagångssättet möjliggjorde skapandet av en variation av data som iterativt kunde utvärderas på hur väl nätverket kände igen olika objekt.

I projektet diskuterades främst två sätt att annotera den insamlade datan, metoderna var avgränsade rektanglar samt semantisk segmentering. På grund av den tidsåtgång som semantisk segmentering medför valdes metoden bort. Istället valdes avgränsande rektanglar med objektklassificering som annoteringsmetod. Rektanglarna, som var ungefär målade på bilderna, omvandlades till listor som innehöll dess koordinater och klasser rektanglarna tillhörde. Drönarens bilderna annoterades även binärt om det ansågs säkert att landa eller inte.





Figur 10: Ett exempel på hur data skiljer sig mellan ett privat och ett offentligt dataset. Till vänster: tagen från Stanfords dataset. Till höger: En godtycklig bild tagen från den privata datamängden [15].

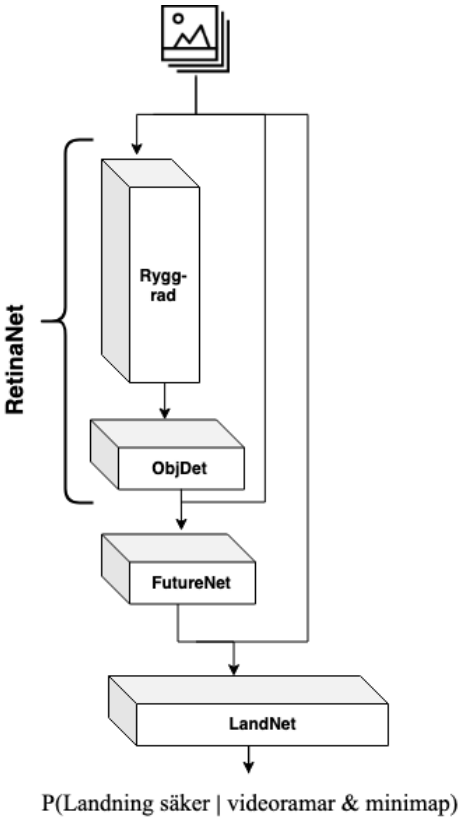
Förbehandling av data

Före data strömmas in i de olika nätverken sker olika sorters förbehandling av datan. Då det finns en stark korrelation mellan förhållandet mellan datamängd och neuroner i nätverket och överanpassning undersöktes möjligheter att utöka datamängden genom att förändra datan på olika sätt [3]. Då annotering av data är en tidskrävande och icke-trivial process och ingen, offentlig färdig-annoterad data användes, kunde endast en begränsad mängd data annoteras. Vid träning av alla modeller användes förändrad samt originell data. Förändringarna involverade bland annat spegling, rotation och förstoring av datan.

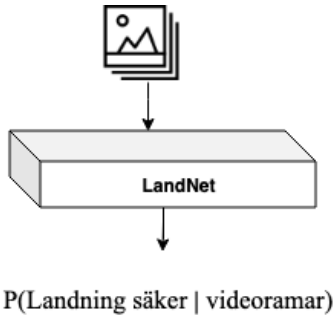
3.4 Modelarkitektur

I arbetet jämfördes två arkitekturer. Den ena applicerade början-till-slut-inlärning och den andra använde ett flertal uppgifts-specialiserade nätverk. I Figur 11 kan en illustration över de båda modellerna ses.

I figur 11 modell (a) kan en se att modellen utgörs av tre mindre modeller. Av de tre var FutureNet och LandNet utvecklat för denna rapport och RetinaNet är utvecklat av Facebooks AI-avdelning. Den första mindre modellen var ett objektdetekterings-nätverk som användes för att producera mini-kartorna. Kartorna var en av inmatningsvariablerna i nästa del. Nästa del var FutureNet. FutureNet användes för att förutspå personers framtida position. Sista delen utgjordes av LandNet som tog emot datan som producerats av FutureNet samt den senast tagna bilden av drönaren. I (b) kan en mycket mindre modell ses. Den mindre modellen var en kopia av LandNet som visas i (a) förrutom att den inte tar emot explicit information utan endast de tre senast tagna drönar-bilderna.



(a) Den mer komplexa modellen låter inmatningsvariabler genomgå fler transformationer.



(b) Ett väldigt simpelt neuralt nätverk som endast utifrån bilder beräknar sannolikhet för säker landning.

Figur 11: Illustration över de två nätverks-arkitekturerna som jämfördes i rapporten.

RetinaNet

RetinaNet är ett objekt-detekterings-nätverk utvecklat av Facebooks AI-avdelning. Att använda RetinaNet som huvudsaklig objekt-detektor var inte uppenbart och det fanns många kandidater för denna uppgift. Ett utdrag av kända objektdetektorer som övervägdes kan ses i figur 12. Valet att använda RetinaNets nätverks-arkitektur togs på grund av dess höga precision och att en öppen implementation av modellen, med en stor mängd dokumentation, fanns tillgänglig[36].

Method	VOC 2007 test	VOC 2012 test	COCO	time (fps)
YOLO	52.7/63.4	57.9/NA	NA	45/155
YOLOv2	78.6	73.4	21.6	40
SSD	77.2/79.8	75.8/78.5	25.1/28.8	46/19
DSSD	81.5	80.0	33.2	5.5
RON	81.3	80.7	27.4	15
RetinaNet	NA	N	39.1	5
RCNN	66	NA	NA	47s
Fast RCNN	77.0	82.3 (wth coco data)	NA	0.5s
Faster RCNN	73.2	70.4	NA	200ms
RFCN	79.5	77.6	29.9	170ms
FPN	NA	NA	36.2	6
Mask RCNN	NA	NA	38.2	2.5

Figur 12: I figuren presenteras ett utdrag av olika, kända, objekt-detekterings-arkitekturer. En kan se att RetinaNet uppnått högst precision, av alla arkitekturer som presenterats, på datamängden COCO [27].

RetinaNet tar emot en bild och beräknar vilka objekt som är närvarande i bilden och var de befinner sig. Nätverket består av två delar, en ryggradsdel och en del som är ansvarig för att att producera avgränsade-rektanglar och att klassificera dess innehåll.

Ryggrads-delen är till för att extrahera värdefulla egenskaper ur bilden. Olika ryggrader är lämpade för olika syften, i detta arbete testat en rad olika [38]. Att retinaNet använder ryggrader för att extrahera egenskaper ur bilder istället för ett arikitektur-unikt tillvägagånssätt innebär att nätverket är väl lämpat för transferering av lärande då många olika för-tränade modeller kan testas. En nackdel till följd av detta är att för-processering av bild-data begränsas till den processering som ryggraden tränats med. RetinaNets arkitektur är uppbyggt på ett sådant sätt att ett byta från en ryggrad till en annan inte är en allt för komplicerad process.

Den andra delen tar emot egenskaperna som inmatningsvariabler och producerar sedan avgränsade, klassificerade, rektanglar. De avgränsande rektanglarna som RetinaNet producerar används sedan som mini-kartor av LandNet.

LandNet

LandNet utvecklades specifikt för detta arbete. Två olika modeller av LandNet användes i arbetet. I den ena modellen tog LandNet emot data som gjorts explicit genom RetinaNet och FutureNet medan den andra tog endast emot tre drönar-bilder tagna i följd. LandNets inmatningsvariabler genomgår en rad transformationer i form av konvolutionella filter. Variablerna, som från början är bilder, ombildas sedan till en en-dimensionell och genomgår ytterligare transformationer i form av ett framåtmatande neuralt nätverk tills den producerar sannolikheten för en säker landning.

FutureNet

FutureNet utvecklades specifikt för detta arbete. FutureNet är ett helt konvolutionellt nätverk. Det innebar att tillplattningen, som beskrevs i LandNet av att modellen plattar ihop bilderna till en en-dimensionell array, aldrig skedde. I FutureNet bibehåller inmatningsbilderna hela tiden sin form, 440x440. Modellen producerade som utmatnings-variabel en bild där nätverket annoterade varje pixel med den förutspådda sannolikheten att en person skulle befinna sig där sex tagna drönar-bilder i framtiden.

Tidsmässig information Vid design av det neurala nätverkets arkitektur planerades det även hur nätverken skulle hantera tidsmässig information. Tidsmässig information kan i detta sammanhang vara hastighet och riktning på människor eller bilar som förflyttar sig genom världen. För att ett neuralt nätverk skall kunna utföra en korrekt bedömning förväntas detta vara ett krav. Detta bör inte ses som en svaghet gällande neurala nätverk då även människor i många fall kräver mer än en statisk videoram för att avgöra om en landning kan ske säkert eller inte. Det fanns fler än ett tillvägagångsätt för att generera tidsmässig information som nätverken kunde nyttja. I figur 13 presenteras ett tillvägagångsätt där tre kontinuerligt tagna bilder kan användas som inmatningsvariabler för nätverket.

Figur 13: Vid undersökning av en enskild bild kan ingen välgrundad slutsats dras gällande personens hastighet och riktning. Om däremot flera bilder undersöks, tagna med konstant tids-mellanrum, framgår tidsmässig och rumslig data som kan leda till en välgrundad slutsats.

3.5 Experiment

I kapitlet presenteras tillvägagångssätt och beskrivningar över hur de olika experimenten utfördes för att besvara frågeställningarna som presenterades i inledningen. I texterna beskrivs inte hur modellerna tränas detaljerat. En mer ingående beskrivning kan finnas i kapitel 3.6.

Experiment 1

För att undersöka om det fanns några direkta fördelar med att göra, vad som ansågs som, värdefull information mer explicit, och med hur hög precision modellerna kunde uppnå, utfördes följande experiment. Två modeller av LandNet och två modeller av FutureNet tränades. Modeller av samma namn hade en identisk intern arkitektur med undantag för inmatningsvariablerna som såg olika ut.

Den ena modellen av LandNet tog in tre stycken drönar-ramar i följd medan den andra modellen tog in den senast tagna drönar-ramen samt en mini-karta över var alla objekt befann sig sex drönar-ramar in i framtiden. Anledningen till att framtida objekts position användes var för att simulera den information som FutureNet förväntades generera till LandNet.

Av FutureNets två modeller tog den ena modellen in tre drönar-ramar, tagna i följd som inmatnings-variabel. Den andra variationen av modellen tog en drönar-ram övertäckt av en färgkodad karta som inmatningsvaraibel. Den färgkodade kartan användes för att synliggöra personers positionsförändring under de sex senast tagna drönar-ramarna. För att beräkna positionsförändringen subtraherades mini-kartan vid tidpunkt t_n med mini-kartan vid tidpunkt t_{n-6}. Båda kartorna är egentligen 2-dimensionella matriser som består utav ettor där personer befinner sig. Differens-matrisen innehöll tre olika siffror efter subtraktionen, -1 om en person befann sig vid en given pixel vid tidpunkt t_n men inte vid t_{n-6}, 0 om en person befann sig vid en given pixel vid tidpunkt t_n och vid t_{n-6} och sist +1 om en person inte befann sig vid en given pixel vid tidpunkt t_n som personen befann sig på vid t_{n-6}. Varje pixel i differens-matrisen färgkodades utifrån vad för siffra pixeln innehöll, alla +1 representeras av färgen grön medan alla -1 representerades av färgen röd. Den färgkodade matrisen målades därefter på drönar-ramen som togs vid tidpunkt t_n och användes som inmatnings-variabel. Modellen tränades endast på att förutspå en persons framtida position.

Vid uppdelning av validerings- och tränings-data delades hela datamängden in i 100 stycken, lika stora delar, där var tionde del flyttades till valideringsmängden och resten till träningsmängden. Båda modellerna tränades och utvärderades på, inbördes, samma datamängd. För utvärdering av modellerna användes modellernas förlustfunktion samt dess precision och återkall. Valideirng utfördes efter varje utförd epok.

Experiment 2

För att undersöka om det gick att applicera överföring av lärande på objekt-detektering testades olika redan existerande ryggradsmodeller. RetinaNet tränades med ryggraderna VGG16, ResNet50, DenseNet121 och MobileNet224. Ryggraderna som användes vid träning valdes framförallt på grund av deras implementationer i den öppna version av RetinaNet som användes i projektet. Det bör även påpekas att samtliga av de fyra ryggraderna har använts i tidigare arbeten och kan anses tillhöra konventionen för neurala nätverk [38]. De olika variationer av RetinaNet tränades mot projektets annoterade datamängd där precisionen av hur väl arkitekturen hittade objekt i en bild jämfördes.

3.6 Träning av modeller

I kapitlet beskrivs de olika metoder och parametrar som användes vid träning av de olika modellerna. Om inget annat specificerats tränades var modell i 50 epoker, varav en epok är 500 uppdateringar.

RetinaNet

Träningen av RetinaNet utfördes ungefär på samma sätt som beskrivet av Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He och Piotr Dollar [26]. Den stora skillnaden involverade hur data förändrades för att utöka storleken på datamängden. Vid träning av det modellen i forsknings-arbetet användes endast spegling som bild-förändring medans i detta arbete används bland annat rotation och förstoring av bilder. RetinaNet använde en ram tagen från drönarens kamera som inmatnings-variabel och producerade två listor, som kombinerades för att skapa kartor, se figur 14 (e). Ena listan innehöll koordinater på var objekt hittades och den andra innehöll av vilka klasser objekten tillhörde. Varje objekt kunde tillhöra tre klasser, människor, bilar eller objekt. Objekt användes som ett grupp-kategori för bland annat stora stenar och träd. Som grund-sanning användes listorna av koordinater och klasser som producerats vid annotering.

RetinaNet tränades parallellt på två olika uppgifter, klassificering och regression. Regressionsuppgiften involverade förutsägelse av koordinater medans klassificering handlade om att kategorisera ett upptäckt objekt. Uppgifterna använde två olika förlustfunktioner, för klassificering användes Focal Loss och för regression Smooth-L1. Funktionerna presenteras nedanför. I funktionerna är y den annoterade sanningen och x är vad nätverket förutspådde.

$$\mathbf{FL}(P_t) = -(1 - P_t)^{\gamma} log(P_t) \qquad \text{(Focal loss)}$$

$$P_t = \begin{cases} P & \text{Om } y = 1 \\ 1 - P & \text{Annars} \end{cases}$$

$$L_{1;\text{smooth}} = \begin{cases} \frac{1}{2}(y - x)^2 & \text{Om } |y - x| < 1 \\ |y - x| - 0.5 & \text{Annars} \end{cases} \qquad \text{(Smooth L1)}$$

LandNet

Vid träning användes "Binary Cross-Entropy", BCE, som förlust-funktion. En annan förlustfunktion som undersöktes var "Mean square error", MSE. Anledningen till att BCE föredrogs beror på hur data annoterades och vad för sorts aktiveringsfuntkion LandNet hade. Då LandNet använde Sigmoid som aktiveringsfunktion för utmatnings-variabeln var LandNets värdemängd [0, 1], kombinerat med att all annoterad data var binär, {0, 1}, hade MSE inneburit ett förlustfunktion som inte straffade felaktiga förutsägelser tillräckligt hårt. BCE däremot bestraffade på en mer passande nivå då den ökar exponentiellt. Funktionerna presenteras nedanför.

$$MSE = \frac{1}{N}\sum_{t=0}^{N}(y_t - x_t)^2 \qquad \text{(Mean Squared Error)}$$

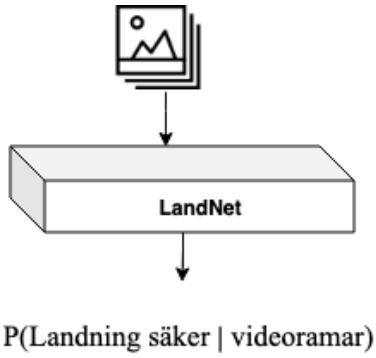
$$BCE = -(y\log(x) + (1-y)\log(1-x)) \qquad \text{(Binary Crossentropy)}$$

I arbetet tränades modellen två gånger med två olika sorters inmatningsvariabler. Den ena modellen tränades med en drönar-bild och en karta över var alla objekt förväntades vara, se figur 14 (b). Den andra tränades med tre drönar-bilder tagna i följd som inmatningsvariabler, se figur 14 (a). Båda tränades med den annoterade sannolikheten för säker landning som grundsanning.

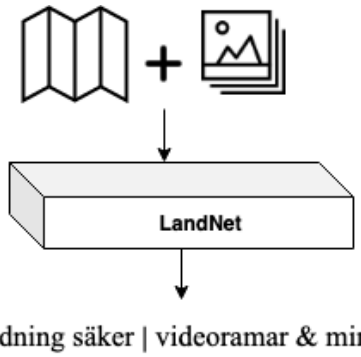
Till optimerare valdes "Adam" [37]. I keras är Stochastic Gradient Descent utgångsvalet av optimerare. Valet att använda "Adam" togs då den har visat sig vara mer förlåtande vid icke-optimerade hyperparametrar och kan därmed ge bättre resultat än Stochastic Gradient Descent vid utveckling av prototyper [31]. För att säkerställa att nätverket inte överanpassas mot tränings-datan utför man en validerings-kontroll efter varje epok. Värdet på hyperparametrarna som vad närvarande vid träning sattes till Keras standardvärden.

FutureNet

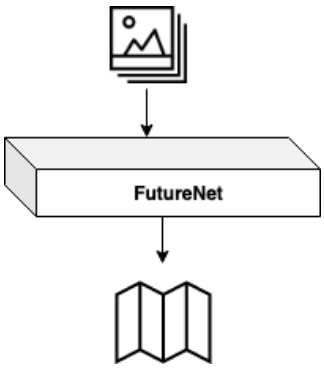
Vid träning av FutureNet användes “Adam“ som optimerare och BCE som förlustfunktion med samma motiveringar som i LandNet. Även samma hyperparametrar användes av FutureNet som av LandNet. FutureNet tränades två gånger med två sorters inmatningsvariabler. Ena modellen tränades med en behandlad drönar-bild som inmatningsvariabel, se figur 14 (d). Behandling utfördes genom att ta en karta över var personer befann sig vid en tidpunkt och sedan subtrahera den med en liknande kartan sex tidsenheter bakåt i tiden. Differensen färg-kodades och applicerades sedan på drönar-ramen för att därefter användes som en inmatnings-variabel. Den andra modellen använde endast tre drönarbilder, tagna i följd, som inmatningsvariabel, se figur 14 (c). Modellerna tränades med den annoterade kartans pixlar, med en storlek på 440x440, sex drönar-bilder in framtiden som grundsanning.



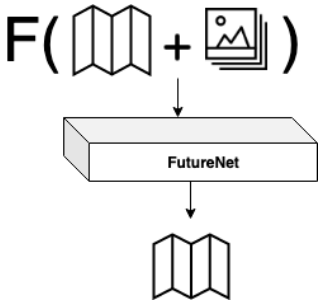
(a) Illustration över in- och utmatningsvariabler som användes av LandNet början-till-slut-version vid träning. Modellen tog endast emot tre bilder som inmatning och producerade sannolikhet som utmatnings-variabel.



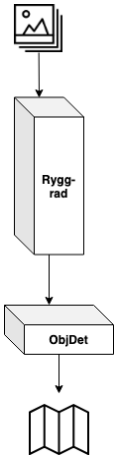
(b) Illustration över hur LandNet tränades och vilka inmatningsvariabler som använts då den använde sig utav explicit data. Modellen tog emot en bild och en karta över objekt.



(c) Den ena modellen av FutureNet tog emot en tre drönar-bilder, tagna i följd, för att förutspå en persons framtida position.



(d) Den andra modellen av FutureNet tog emot en bild som färgkodats med hjälp av kartor över var objekt befann sig.



(e) Illustration över hur RetinaNet tränades och vilka inmatningsvariabler som använts.

Figur 14: Illustration över inmatnings- och utmatningsvariabler som användes vid träning av olika modeller. I (d) skall funktions-annotering illustrerar att bilderna och kartorna genomgår en behandling före de används som inmatnings-variabel.

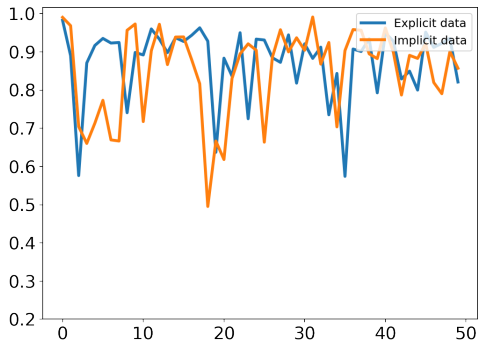
4

Resultat & Diskussion

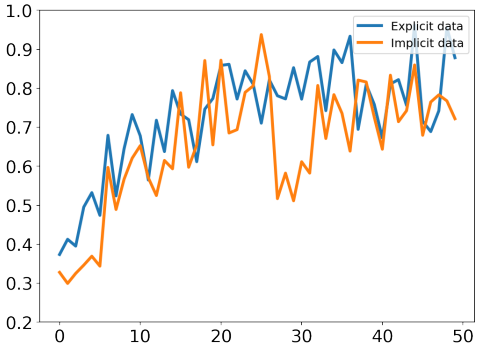
I kapitlet presenteras de resultaten som extraherades vid träning och utvärdering av modellerna. Begreppen explicit och implicit information används genomgående i kapitlet. Explicit information innebär att värdefull information i drönar-ramar har extraherats och konkretiserats för att sedan användas som inmatningsvariabler. I implicit information används endast drönar-ramars råa bild-information som inmatningsvariabler.

LandNet

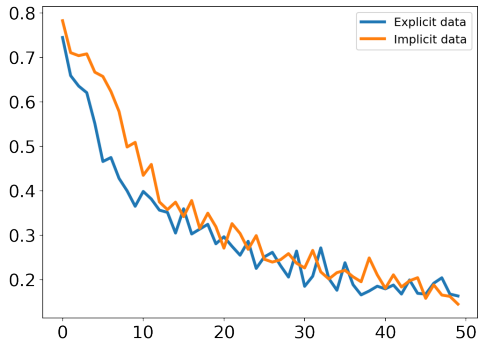
Tränings och valideringsresultaten av LandNet som utvecklades specifikt för detta arbete presenteras i figur 15. De olika färgade linjerna visar hur resultaten skiljer sig beroende på modellens inmatningsvariabel.



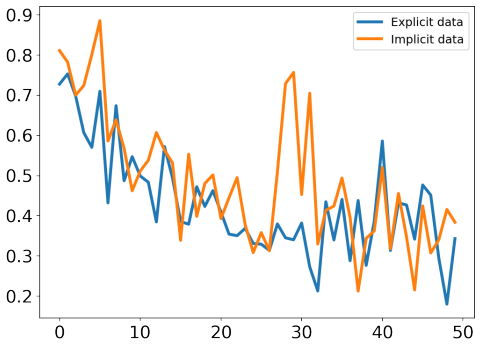
(a) Y-axeln visar återkallelse vid utvärdering på valideringsmängd. Högre är bättre.



(b) Y-axeln visar precision vid utvärdering på valideringsmängd. Högre är bättre.



(c) Y-axeln visar storlek på förlustfunktionen vid träning på träningsmängden. Lägre värde är bättre.



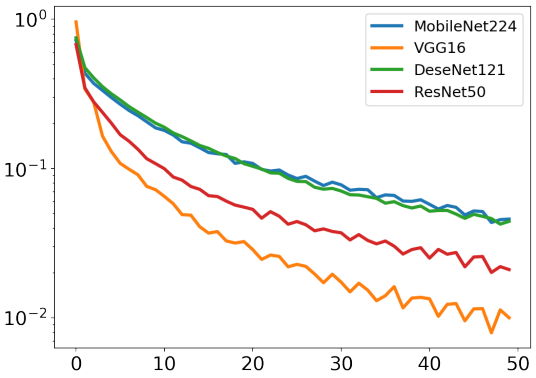
(d) Y-axeln visar storleken på förlustfunktionen vid utvärdering på valideringsmängden. Lägre värde är bättre.

Figur 15: x-axeln på graferna visar antal epoker och y-axeln specificeras under respektive graf.

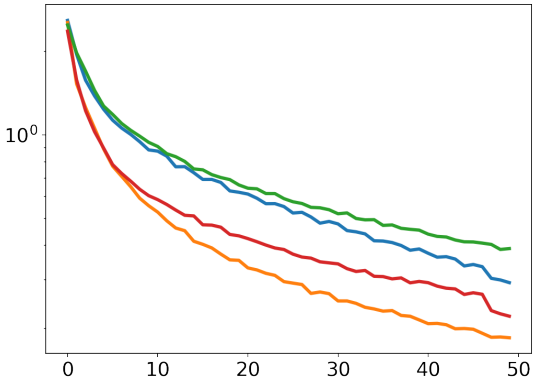
Den låga precisionen och högre återkallelsen som visas i graferna i figur 15 påvisar att båda modellerna är mycket aggressiva i början. Modellerna bedömer felaktigt att en landning är säker att genomföras när det inte är fallet. Efter mer träning ökar precisionen trots att återkallelsen, om något volatil, bibehåller ett högt värde.

RetinaNet

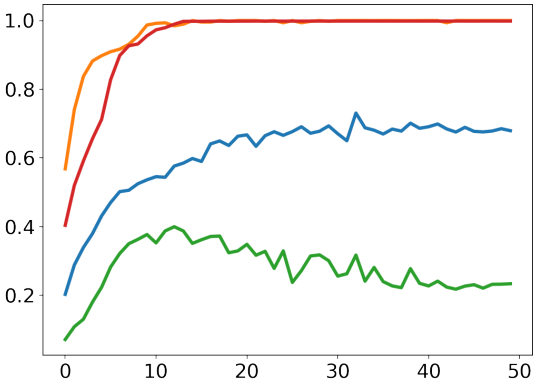
Nedanför presenteras resultaten av träning och validering av RetinaNet. De olika färgade linjerna visar hur resultaten skiljer sig beroende på modellens olika ryggrader.



(a) Y-axeln visar storlek på förlustfunktionen av Klassificeringen. Lägre värde är bättre. Logaritmisk skala för enklare jämförelse.



(b) Y-axeln Storlek på förlustfunktionen av RetinaNets regressions-uppgift. Lägre värde är bättre. Logaritmisk skala för enklare jämförelse.



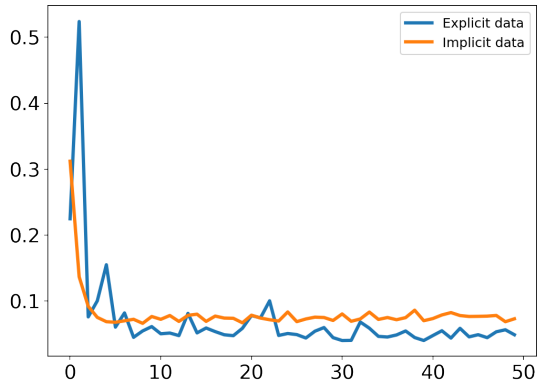
(c) Y-axeln visar mAP vid utvärdering på valideringsdata. högre värde är bättre.

Figur 16: x-axeln på graferna visar antal epoker och y-axeln specificeras under respektive graf.

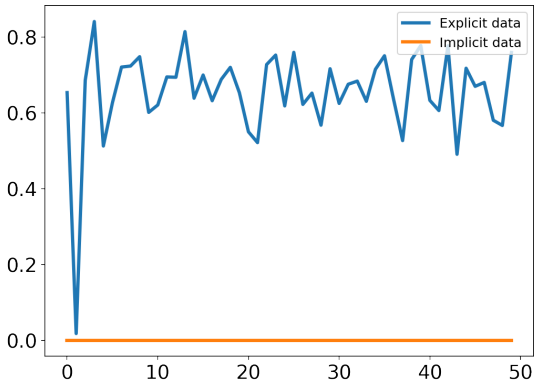
Graferna i Figur 16 påvisade att ResNet50 och VGG16 fick högst topp-precision av alla ryggrader som testades. Då VGG16 och ResNet50 har flest parametrar av de ryggrader som testats är resultatet därmed inte överraskande. I figuren kan en även se att DenseNet påvisar lägst topp-precision, detta är överraskande då DenseNet har fler parametrar än MobileNet. DenseNets precision-kurva når en topp tidigt och sjunker sedan. Det kan vara ett tecken på överanppassning. Det är märkligt att dessa tecken endast syns på DenseNet medans ResNet50 och VGG16 inte gör det. Att DenseNet har lägst topp-precision måste inte bero på överanpassning utan kan bero på felaktig implementering eller att den fastnade i ett lokalt minima.

FutureNet

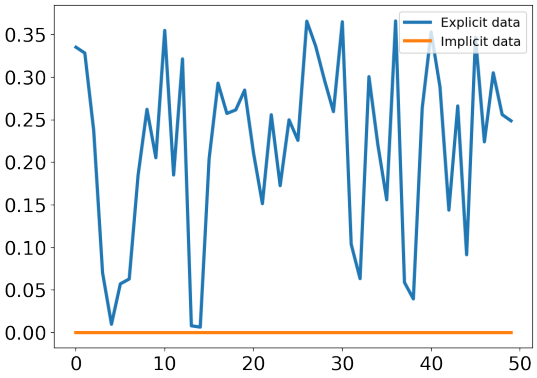
Tränings-och valideringsresultaten av futureNet som utvecklades specifikt för detta arbete presenteras i figur 17. De olika färgade linjerna visar hur resultaten skiljer sig beroende på modellens inmatningsvaraibel.



(a) Y-axeln visar storleken av förlustfunktion vid utvärdering på valideringsmängden. Lägre värde är bättre.



(b) Y-axeln visar precision vid utvärdering på valideringsmängden. Högre är bättre.



(c) Y-axeln visar Återkallelse vid utvärdering på valideringsmängden.

Figur 17: X-axeln på graferna visar antal epoker och y-axeln specificeras under respektive graf.

Graferna i figur 17 kan få en att uppfatta skillnaden på förlust-funktionens storlek mellan de två modellerna som försumbar. Då uppgiften handlar om att klassificera pixlar i en 440x440 stor bild, varav den stora majoriteten av pixlar är nollor, inleder förmodligen båda modellerna med att justera vikterna sådan att varje pixel klassificeras som en nolla oavsett inmatningsvariabler. En kan sedan se att modellen som tränas med explicit data verkar ha bättre förståelse för uppgiften och presterar därmed kontinuerligt bättre med högre topp-precision än den andra modellen. En överraskande del av resultatet är den höga volatilitet den explicita modellens återkallelse visar i kontrast med den mer stabila precision modellen kontinuerligt uppvisar.

Felkällor

De två felkällor som upptäcktes i projektet var bildkvalitén som kom från drönarens kamera samt svårigheter i konsekvent annotering.

Kamera Då den inköpta drönarens kamera fick monteras ur sin position och ett fisköga monterades ovanpå kameran uppstod en del kompromisser med bildkvalitén. Kameran blev inte lika stabil då den inte längre kunde hålla sin position lika perfekt som när den var integrerad i drönaren. Fiskögat hade samma problem, där vind och andra yttre faktorer gjorde så fiskögat rörde på sig. Av dessa två anledningar var inhämtningen av bilden inkonsekvent. Kameran bör också ha något högre upplösning än den som följer med drönaren Tello.

Annotering Ett problem som inte tidigare var känt var hur svårt det var att vara konsekvent vid annotering av data. Både när det gäller var objekt börjar och slutar i avgränsande rektanglar och annotering kring om det anses vara säkert att landa eller inte. När sker brytpunkten mellan säker landning eller inte? Hur konsekvent är den annotering som vår datamängd innehåller?

Hållbar utveckling och etik Syftet med projektet var att undersöka hur en kan överföra ansvaret för riskbedömningen, om huruvida en drönare kan säkert landa, från människan till drönaren genom ett artificiellt neuralt nätverk. En lyckad överföring av denna riskbedömning kan bidra till en högre säkerhet då det kan uppkomma situationer där operatörer förlorar kontakt med drönare. Istället för att drönaren påbörjar en landning direkt vid förlorad kontakt kan den istället utföra en riskbedömning och välja det handlingsalternativ som är mest lämpad för situationen. Därmed kan skador på människor och annan egendom undvikas.

5

Slutsats

Syftet med projektet var att undersöka hur en kan överföra ansvaret för riskbedömningen, om huruvida en drönare kan säkert landa, från människan till drönaren. Resultaten indikerar att ett neuralt nätverk kan vara en gångbar lösning i att utföra riskbedömningar om huruvida en säker landning kan utföras. Detta argumenteras för då modellerna, som inte bör anses som fullt optimerade, uppnådde en topp-precision på 0.93. Arbetet påvisar även att det finns tydliga skillnader i både precision och återkallelse vid träning och utvärdering av modeller som använder explicit kontra implicit data. Hur stora skillnaderna är beror till hög grad på vad för sorts uppgift som nätverket tränas på att utföra.

FutureNet, ett helt konvolutionellt neuralt nätverk, som tränats på att förutspå en persons framtida position visade, enligt oss, medelmåttiga resultat då den endast kunde identifiera 35 % av pixlarna där en person befann sig i framtiden korrekt. Huruvida ett helt konvolutionellt neuralt nätverk kan förutspå en persons framtida position är utifrån resultaten svårt att besvara då det finns flera sätt att optimera modellen på så som fler epoker av träning, större datamängd och mer optimerad modell-arkitektur.

Objektdetekterings-modellen påvisade goda resultat. Möjligheten att applicera transferering av lärande på en så vitt skild domän är utan tvekan möjlig. Det var även tydligt att vissa ryggrader var mer lämpade än andra. Enligt våra experiment hade ResNet och VGG högst precision och uppnåde bästa topp-resultat av de fyra.

5.1 Möjligt fortsatt arbete

Nedan listas de främsta punkter som vi anser vara de viktigaste för fortsatt arbete.

Mer data

Ett stort problem i ett mindre projekt som använder sig av en egeninsamlad datamängd, är att storleken på datamängden inte är i önskvärd skala. Vid fortsatt arbete bör mer data samlas in för att öka precisionen på nätverket och dess generaliserade egenskaper [33]. En del-lösning till en större datamängd är att samla in bilder och använda objektdetekteringen för att automatiskt annotera objekts positioner och dess objektsklass. Detta möjliggör träningsdata till nätverk så som FutureNet som inte kräver någon mer annotering.

Förbättrad bildkvalité

Önskvärt hade varit att fisk-ögat redan var integrerad drönardesignen med en bättre kamera eller någon liknande lösning som hade bidragit till en stabilare och bättre bild.

Fler uppgifts-specifika nätverk

För att skapa ett robust nätverket föreslår vi fler uppgifts-specifika nätverk som konkretiserar mer information i en bild. Då fler konkreta separata mönster tas ut i en bild ges större chans till nätverket att gissa rätt. Ett exempel på ett uppgifts-specifikt nätverk vi hade velat implementera är ett nätverk som endast försöker förstå hur långt bort objekt i bilder befinner sig.

Objekt-spårning

Idag är nätverksarkitekturen utformad att hitta var i bilden som objekt befinner sig och utefter dessa objekt förutspås var de kommer vara om sex stycken bildramar i framtiden. Dock särskiljs inte objekt som överlappar varandra utan överlappande objekt skickas till futureNet som om det vore ett enda objekt. Detta kan skapa problem då futureNet försöker förutspå var de kommer att vara. Vi föreslår att hålla koll på enskilda objekt och särskilja dessa till futureNet.

Semantisk segmentering

Metoden valdes bort då den tidsåtgång som krävs för att annotera bilder med semantisk segmentering är väldigt tidskrävande. Däremot bör metoden testas i ett fortsatt arbete och jämföras mot avgränsande rektanglar. Informationen blir mycket mer exakt var objekt befinner sig vilket troligen är mer värdefullt för ett nätverk att lära sig på.

Referenser

[1] J. Brownlee, "Supervised and Unsupervised Machine Learning Algorithms ", Machine Learning Mastery, 2019. [Online]. Tillgänglig: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/. [Hämtad: 04- Apr- 2019].

[2] "What is end-to-end deep learning? - ML Strategy (2) | Coursera ", Coursera, 2019. [Online]. Tillgänglig: https://www.coursera.org/lecture/machine-learning-projects/what-is-end-to-end-deep-learning-k0Klk. [Hämtad: 04- Apr- 2019].

[3] B. Mehlig, "Artifical Neural Networks", Arxiv.org, 2019. [Online]. Tillgänglig: https://arxiv.org/pdf/1901.05639.pdf. [Hämtad: 07- May - 2019].

[4] J. Brownlee, "A Gentle Introduction to Transfer Learning for Deep Learning ", Machine Learning Mastery, 2019. [Online]. Tillgänglig: https://machinelearningmastery.com/transfer-learning-for-deep-learning/. [Hämtad: 04- Apr- 2019].

[5] Arxiv.org, 2019. [Online]. Tillgänglig: https://arxiv.org/pdf/1502.04042.pdf. [Hämtad: 04- Apr- 2019].

[6] "Build with AI | DeepAI ", DeepAI, 2019. [Online]. Tillgänglig: https://deepai.org/machine-learning-glossary-and-terms/activation-function. [Hämtad: 04- Apr- 2019].

[7] Arxiv.org, 2019. [Online]. Tillgänglig: https://arxiv.org/pdf/1812.03079.pdf. [Hämtad: 04- Apr- 2019].

[8] "Build with AI | DeepAI ", DeepAI, 2019. [Online]. Tillgänglig: https://deepai.org/machine-learning-glossary-and-terms/loss-function. [Hämtad: 04- Apr- 2019].

[9] "Goodfellow, Ian; Bengio, Yoshua; Courville, Aaaron (2016) Deep Learning. MIT Press. p. 196. ISBN 9780262035613 "

[10] "What Even is Computer Vision? ", Towards Data Science, 2019. [Online]. Tillgänglig: https://towardsdatascience.com/what-even-is-computer-vision-531e4f07d7d0. [Hämtad: 09- Apr- 2019].

[11] Stanford.io, 2019. [Online]. Tillgänglig: https://stanford.io/2TEpCCv. [Hämtad: 09- Apr- 2019].

[12] Arxiv.org, 2019. [Online]. Tillgänglig: https://arxiv.org/pdf/1404.7828.pdf. [Hämtad: 09- Apr- 2019].

[13] Store.dji.com, 2019. [Online]. Tillgänglig: https://store.dji.com/product/tello?vid=38421. [Hämtad: 09- Apr- 2019].

[14] "Tello ", Ryzerobotics.com, 2019. [Online]. Tillgänglig: https://www.ryzerobotics.com/tello. [Hämtad: 09- Apr- 2019].

[15] "Computational Vision and Geometry Lab ", Stanford.io, 2019. [Online]. Tillgänglig: https://stanford.io/2WEm4lE. [Hämtad: 09- Apr- 2019].

[16] "DeepLearning series: Convolutional Neural Networks ", Medium, 2019. [Online]. Tillgänglig: https://medium.com/machine-learning-bites/deeplearning-series-convolutional-neural-networks-a9c2f2ee1524. [Hämtad: 09- Apr- 2019].

[17] "NVIDIA cuDNN ", NVIDIA Developer, 2019. [Online]. Tillgänglig: https://developer.nvidia.com/cudnn. [Hämtad: 09- Apr- 2019].

[18] "CUDA Zone ", NVIDIA Developer, 2019. [Online]. Tillgänglig: https://developer.nvidia.com/cuda-zone. [Hämtad: 09- Apr- 2019].

[19] Beta.techcrunch.com, 2019. [Online]. Tillgänglig: https://beta.techcrunch.com/wp-content/uploads/2015/11/nvidia-tesla-accelerated-performance.png. [Hämtad: 09- Apr- 2019].

[20] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", arXiv.org, 2019. [Online]. Tillgänglig: https://arxiv.org/abs/1506.01497. [Hämtad: 09- Apr- 2019].

[21] G. Stein, T. Learning and G. Stein, "The Importance of Simulation in the Age of Deep Learning", Cachestocaches.com, 2019. [Online]. Tillgänglig: http://cachestocaches.com/2018/12/importance-simulation-age-deep-learning/. [Hämtad: 09- Apr- 2019].

[22] Arxiv.org, 2019. [Online]. Tillgänglig: https://arxiv.org/pdf/1612.07828.pdf. [Hämtad: 09- Apr- 2019].

[23] "TensorFlow", TensorFlow, 2019. [Online]. Tillgänglig: https://www.tensorflow.org/. [Hämtad: 09- Apr- 2019].

[24] "Home - Keras Documentation", Keras.io, 2019. [Online]. Tillgänglig: https://keras.io/. [Hämtad: 09- Apr- 2019].

[25] “Deep Learning Framework Power Scores 2018“, Towards Data Science, 2019. [Online]. Tillgänglig: https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a. [Hämtad: 09- Apr- 2019].

[26] T. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, Focal Loss for Dense Object Detection", arXiv.org, 2019. [Online]. Tillgänglig: https://arxiv.org/abs/1708.02002. [Hämtad: 16- Apr- 2019].

[27] Zsc.github.io, 2019. [Online]. Tillgänglig: https://zsc.github.io/megvii-pku-dl-course/slides/Lecture6(Object%20Detection).pdf. [Hämtad: 25- Apr- 2019].

[28] S. Daily, "Zoox Self-Driving with Ethan Dreyfuss - Software Engineering Daily", Software Engineering Daily, 2019. [Online]. Tillgänglig: https://softwareengineeringdaily.com/2019/02/20/zoox-self-driving-with-ethan-dreyfuss/. [Hämtad: 25- Apr- 2019].

[29] “Breaking down Mean Average Precision (mAP)“, Towards Data Science, 2019. [Online]. Tillgänglig: https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52. [Hämtad: 25- Apr- 2019].

[30] Arxiv.org, 2019. [Online]. Tillgänglig: https://arxiv.org/pdf/1612.07828.pdf. [Hämtad: 25- Apr- 2019].

[31] A. Karpathy, “A Recipe for Training Neural Networks“, Karpathy.github.io, 2019. [Online]. Tillgänglig: http://karpathy.github.io/2019/04/25/recipe/?fbclid=IwAR05FxWQVKeTiy0VbQVib_7y6sgfy6OBPTih2Kf0LPOHm3yc8nMOlja5gV8. [Hämtad: 26- Apr- 2019].

[32] “AI Can Predict the Future Location of Vehicles - NVIDIA Developer News Center“, NVIDIA Developer News Center, 2019. [Online]. Tillgänglig: https://news.developer.nvidia.com/ai-can-predict-the-future-location-of-vehicles/. [Hämtad: 07- May- 2019].

[33] Static.googleusercontent.com, 2019. [Online]. Tillgänglig: https://static.googleusercontent.com/media/research.google.com/sv//pubs/archive/35179.pdf. [Hämtad: 07- May- 2019].

[34] “Stanford University CS231n: Convolutional Neural Networks for Visual Recognition“, Cs231n.stanford.edu, 2019. [Online]. Tillgänglig: http://cs231n.stanford.edu/index.html. [Hämtad: 09- May- 2019].

[35] "Take a Spin with World's Fastest GPU Accelerator – for Free | NVIDIA Blog", The Official NVIDIA Blog, 2019. [Online]. Tillgänglig: https://blogs.nvidia.com/blog/2015/01/28/test-fastest-accelerator/. [Hämtad: 09- May- 2019].

[36] 2019. [Online]. Tillgänglig: https://github.com/fizyr/keras-retinanet. [Hämtad: 20- May- 2019].

[37] Arxiv.org, 2015. [Online]. Tillgänglig: https://arxiv.org/pdf/1412.6980.pdf. [Hämtad: 29- May- 2019].

[38] Arxiv.org, 2018. [Online]. Tillgänglig: https://arxiv.org/pdf/1810.00736.pdf. [Hämtad: 29- May- 2019].