Djupinlärning för identifikation av föremål i bilder

Jan Söderberg 40531
Kandidatavhandling i datavetenskap
Handledare: Marina Waldén
Fakulteten för naturvetenskaper och teknik
Åbo Akademi
2019

Innehåll

1 Inledning

Maskininlärning är en term som uppkommer i litteraturen redan på 50-talet. Enligt Arthur Samuel kan maskininlärning tänkas som att programmera en dator att bete sig på ett sätt som vi för människor eller djur skulle säga att involverade någon sorts inlärning. [1]

Många olika tekniker för att nå detta ändamål har använts genom åren med varierande framgång. I dagens läge används maskininlärningsalgoritmer i flera olika produkter och tillämpningar. Stora framsteg har gjorts inom bland annat bild- och talanalys. Ökad beräkningsbarhet av hårdvara och tillgängligheten av stora mängder data har möjliggjort återuppvaknandet av tidigare tekniker som inte i sin begynnelse åstadkommit någon större succe. [2]

Traditionellt har tillverkningen av maskininlärningsalgoritmer behövt mycket manuellt arbete och expertis inom tillämpningsområdet för att kunna åstadkomma bra resultat. Under de senaste åren har djupinlärning möjliggjort att maskininlärningsalgoritmer inte längre behöver samma mängd manuellt arbete för att åstadkomma samma resultat eller även bättre resultat. Genom representionella inlärningsmetoder kan en algoritm själv bestämma vilka särdrag i datan påverkar på hur datat klassifieras – något som tidigare gjorts manuellt. [2]

Föremålsidentifiering är ett centralt problem inom datorseende – ett forskningsområde inom maskininlärning. Där människor genast kan identifiera olika föremål och hur de interagerar med varandra har detta varit en betydlig utmaning för datorsystem [2]. Djupinlärningsmetoder har möjliggjort stora framsteg inom området.

Syftet med kandidatavhandlingen är att presentera och jämföra olika tekniker för föremålsidentifiering med hjälp av djupinlärningsmetoder. I de första två kapitlen beskrivs bakgrundsinformation för övervakade maskininlärningstekniker som används i de senare kapitlen. I kapitel 4 beskrivs olika föremålsidentifierings metoder i detalj och kapitel 5 jämför dessa. Målet är att ge en helhetsbild över vad som kan göras i dagens läge med dessa system, vad deras svagheter är och hur de presterar.

2 Övervakad inlärning

Den mest använda maskininlärningsmetoden i dagens läge kallas för övervakad inlärning. En inlärningsalgoritm av denna typ har tillgång till data som för varje datapunkt har märkts ut det värde vi vill att den slutliga modellen skall förutspå för liknande datapunkter. [2]

Matematiskt sätt kan en sådan algoritm tänkas som en funktions uppskattare för en funktion f med tillgång till både input värden x och output värdet y, men utan någon ex-

plicit formel för f. Algoritmens uppgift är att hitta en lämplig funktion för att så bra som möjligt uppskatta $f(x)$ för datapunkterna som algoritmen har tillgång till vid träningsskedet. För att nå detta ändamål används vad som kallas en förlustfunktion som beräknar hur mycket det uppskattade värdet skiljer från det verkliga. Algoritmens mål är att minimera förlustfunktionen. På detta sätt åstadkommer vi en funktion som i det optimala scenariot uppskattar rätta värden även för datapunkter som algoritmen aldrig tidigare sett. [2]

I träningsskedet matas systemet in med träningsexempel – data som märkts ut med det värde som vi vill att modellen skall förutspå liknande data som. Algoritmen beräknar skillnaden mellan uppskattningarna och de egentliga värdena. Sedan modifieras modellens interna parametrar för att åstadkomma en bättre uppskattning i framtiden. [2]

Övervakade inlärningsalgoritmer kan i allmänhet delas in i två olika kategorier - regression och klassifiering. Regressionsalgoritmer förutspår en funktion med ett numeriskt output värde $x \in \mathbb{R}$ medan klassifieringsalgoritmer förutspår en funktion som kategoriserar indata till en av k kategorier $x \in \{1, ..., k\}$. [2]

2.1 Gradient Descent

Gradient descent är en metod som används i träningsskedet av flera övervakade inlärningsalgoritmer. Algoritmens uppgift är alltså att minimera förlustfunktionen med hjälp av att modifiera modellens interna parametrar. [3]

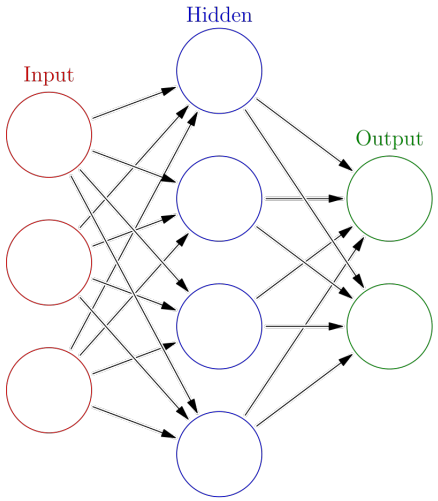
Gradienten av en funktion pekar i riktningen där funktionen har sin största förökningstakt medan den negativa gradienten pekar i riktningen där funktionen har sin största förminskningstakt. Genom att iterativt kalkylera den negativa gradienten av förlustfunktionen och updatera parametrarna hittar vi ett värde för parametrarna som så bra som möjligt minimerar funktionens värde. Algoritmen börjar med något värde på parametrarna θ och updaterar dessa iterativt på följande sätt:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} C(\theta) \tag{1}$$

α kallas inlärningstakten och styr hur stora steg algoritmen tar vid varje iteration.

Stochastic Gradient Descent(SGD) är en populär variation på Gradient Descent algoritmen. Hur den skilljer sig ifrån vanlig Gradient Descent är hur mycket data som används för att beräkna förlustfunktionen för varje uppdatering av parametrarna θ. Vanlig Gradient Descent, även kallad Batch Gradient Descent, räknar ut gradienten för fölustfunktionen med hjälp av hela träningsmängden, vilket för stora mängder data är långsamt. SGD uppdaterar parametrarna θ för varje träningsexempel i träningsmängden skillt. [3]

(skriv bättre)



Figur 1: Ett neuralt nätverk visualiserad som en viktad graf.

2.2 Artificiella Neurala Nätverk

Många av de mest lyckade maskininlärningsalgoritmerna som används i dagens läge baserar sig på artificiella neurala nätverk(ANN). Dessa är basen för djupinlärningsalgoritmerna. [2]

Motivationen bakom terminologin härstammar från det ursprungliga målet av att modellera biologiska neurala nätverk hos människan, d.v.s. neuroner och synapser i människohjärnan. De moderna tillämpningarna av ANN har ändå väldigt lite likhet med biologiska processer utanför terminologin. [4] Neurala nätverk är ramverk för övervakade inlärningsalgoritmer som istället för att uppskatta en enda funktion, kan lära sig att uppskatta mera komplexa funktioner genom att sätta ihop och uppskatta flera olika funktioner som alla tillsammans fungerar för samma ändamål. [2]

Ett ANN kan tänkas som en viktad graf av artificiella neuroner som är organiserade i olika lager. Första lagret kallas för input lagret och det sista för output lagret (se figur 1). Arkitekturen av ett ANN kan också innehålla ett eller flera gömda lager. Om ett ANN har flera än ett gömt lager kallas den för ett djupt ANN. [2]

Informationsflödet genom nätverket går från input lagret till outputlagret genom de gömda lagren. Lagren i nätverket kan innehålla olika antal av neuroner. Varje lager förutom input lagret har också en extra neuron som kallas bias.

En neurons input fås genom att multiplicera den föregående lagrets output med vikten till neuronen och addera ihop alla dessa för alla inkommande vikter tillsammans med bias värdet. En aktiveringsfunktion körs på det här värdet vilket producerar neuronens output.

(skriv mera)

2.3 Bakåtpropagering

Bakåtpropagering är en algoritm som används för träning av artificiella neurala nätverk. Algoritmen beräknar partiella derivatorna av förlustfunktionen i förhållande till varje vikt och bias - $\frac{\partial C}{\partial w_{jk}^{[j]}}$ och $\frac{\partial C}{\partial b_j^{[l]}}$. Med andra ord beräknar algoritmen ett mått på hur mycket förlustfunktionen förändras då en vikt eller bias förändras. Med hjälp av partiella derivatorna kan Gradient Descent användas för att förminska förlustfunktionen. [5]

(skriv mera)

3 Faltningsnätverk

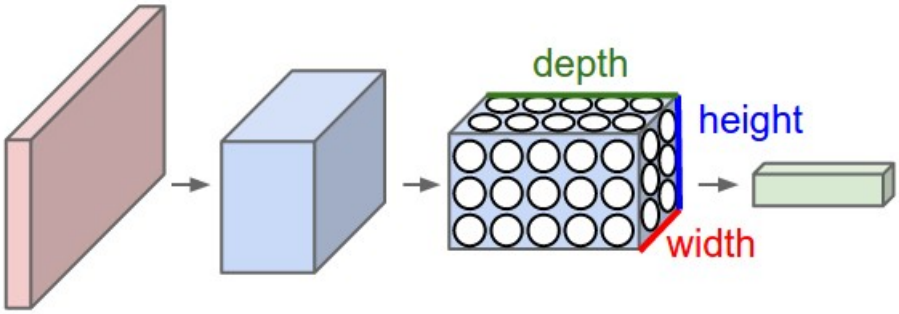
Faltningsnätverk är djupa neurala nätverk som introducerades av Yann Lecun år 1998. De har haft mycket framgång i bland annat bild och videoanalys [2]. Dessa är basen på de moderna metoderna för föremålsidentifiering. Strukturen bygger på neurala nätverken från tidigare, men faltningsnätverken gör antagandet att indata till nätverket är en bild eller något som kan kodas som en bild. Det här antagandet gör att arkitekturen på nätverket kan optimeras på ett vettigt sätt för ändamålet. [4] Ett normalt neuralt nätverk med bara fullt-kopplade lager lämpar sig inte bra till bilddata med större resolution. Mängden kopplingar i nätverket blir fort svårbehandlad då resolutionen på bilden ökar. Till skillnad från neurala nätverken från tidigare är inte alla neuroner kopplade till varandra i successiva lager vilket förminskar beräkningsbelastningen. [4]

Ett faltningsnätverk består av flera olika typs lager: faltningslager, pooling lager och fullt kopplade lager. I ett typiskt faltningsnätverk består de första lagrerna av faltningslager och pooling lager. De sista lagrerna är fullt kopplade lager. [4]

Neuronerna i faltningsnätverken är arrangerade i volymer. Om indata till nätverket är en bild då är dimensionerna t.ex. längd, höjd och djup(RGB kanalerna). Faltningslagren extraherar särdrag från dess input som sedan skickas vidare och transformeras till mera abstrakta representationer av bilden. Genom de olika lagren i nätverken transformeras bilden till det sista fullt kopplade lagret som representerar t.ex. de olika klasserna för bild klassifiering. Figur 2 visar hur ett faltningsnätverk transformerar data från höjd- och längdled till en större volym i djupled. [4]

3.1 Faltningslager

Ett faltningslager består av vad som kallas filter. Dessa är parametrarna som faltningsnätverket lär sig vid träning av nätverket. Filtren är mindre i höjd och längd dimensionerna än indatan, men har samma djup. Ett faltningslager har typiskt många filter som alla är känsliga till olika sorters mönster i bilden.



Figur 2: Faltningsnätverk

För indata till ett faltningslager tas skalärprodukten mellan lagrets filter skillt i varje position av indatat, vilket för varje enskilld filter ger en tvådimensionell output i höjd- och längdled. Alla dessa sätts ihop i djupled och skickas som input till nästa lager. Volymen som ett falningsnätverk producerar kallas en aktiveringskarta. Genom att ändra på antalet filter som används kan volymen på aktiveringskartan ändras. Idén bakom att använda filter i på varandra följande lager är att de tidigare lagren lär sig att känna igen enkla särdrag i bilden som t.ex. kanter, medan de senare lagren lär sig att känna igen mer komplexa mönster från de tidigare lagren.

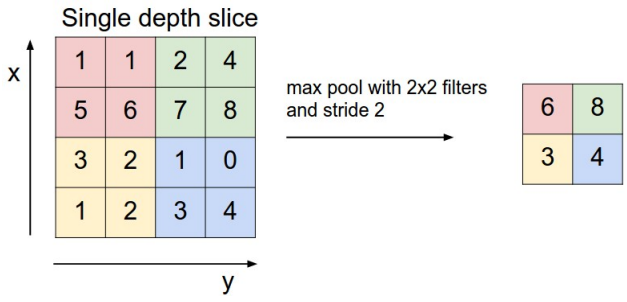
Det finns några modifikationer som kan göras till ett faltningslager. Man kan t.ex. istället för att räkna skalärprodukten mellan filtren och varje position av indatat bara göra det för en mindre del. Om man tänker att filtret glider genom inputvolymen pixel per pixel, kan vi ändra på hur många pixlar i taget som glidningen sker på. En annan modifikation som kan göras är är att fylla kanterna av inputvolymen med nollor. Detta möjliggör att man kan kontrollera hur stort utdatat är i höjd- och längdled vilket är nödvändigt t.ex. då man vill ha samma dimensioner för indata och utdata. [4]

3.2 Övriga lager

Poolinglager är lager som vanligtvis läggs mellan faltningslagren. Syftet med ett poolinglager är att krympa dimensionen av data i höjd- och längdled. På detta sätt förminskas antalet parametrar i nätverket och genom det beräkningskomplexiteten. Detta fungerar även som ett sätt att förminska överanpassning av modellen. [4]

Figur 3 visar hur MAX-pooling fungerar med 2x2 filter med en (stride?) av 2 på indata med dimensionerna 4x4.

De sista lagrerna i nätverket är fullt kopplade lager som ger den slutliga klassifieringen. Helt som i de normala neurala nätverken är alla neuronerna i dessa lager kopplade till alla andra neuroner i de föregående lagrerna.



Figur 3: Max pooling

4 Identifiering av föremål i bilder

Föremålsidentifiering i bilder kan delas in i flera olika nivåer av identifieringsproblem. I det enklaste fallet har vi en bild som vi vill klassifiera i någon kategori utan att ta hänsyn till vart i bilden föremålet ligger enligt vilket bilden klassifieras. Nästa nivå är att förutom klassifiering också lokalisera föremålet i bilden d.v.s. hitta en rektangulär area inom bilden som omramar föremålet så bra som möjligt. Detta utvidgas naturligt till flera föremål inom en bild. Den här avhandlingen fokuserar på klassifiering och lokalisering av flera föremål.

Ett typiskt exempel på hur en klassifiering och lokalisering algoritm kan visualiseras är att den för en bild eller video ritar en rektangel runt de identifierade föremålen och med text beskriver till vilken kategori föremålet tillhör. Vanligtvis ges också en sannolikhetsmått på hur säker modellen är på att rutan faktist innehåller föremålet [6]. Med bilddata som input skall algoritmen alltså ge en mängd x och y kordinater samt längd och höjd på ett område innanför bilden som omramar föremålen så bra som möjligt. Alla dessa områden klassifieras sedan i kategorier.

Moderna metoder som använder djupinlärning för att lösa problemet kan delas in i två olika kategorier - två-stegs-metoder och ett-stegs-metoder. Två stegs metoder försöker först hitta relevanta regioner inom bilden som sedan klassifieras skillt, medan ett-stegs-metoder rakt kan förutspå klasser och regioner i bilden med hjälp av faltningsnätverk [6]. Två stegs metoder har i allmänhet haft bättre noggrannhet med jämförelse till ett-stegs-metoder. Ett stegs metoder har däremot fungerat snabbare men med sämre noggranhet än två-stegs-metoderna. [7]

I det första fallet kan man använda faltningsnätverk för att klassifiera olika delar av bilden. Problemet förblir att hitta lämpliga delregioner av bilden så att algoritmen är beräkningsbart effektivt. Eftersom ett föremål kan ligga var som helst i bilden och i vilken storlek som helst måste vi ha ett sätt för att hitta dessa möjliga delregioner utan att söka igenom alla möjliga sådana. [6]

Eftersom algoritmerna baserar sig på övervakad inlärning, måste data som används i

Figur 4: Visualisering av IoU som mått för överlappande begränsningsområden.

träningsskedet vara märkt med rätta svar. För att kunna evaluera hur bra en uppskattad begränsningsområde överensstämmer med de märkta svaren räcker det inte med ett enda diskret värde så som vid enkel bildklassifiering. Det här är på grund av att det är osannolikt att även en väl presterande model förutspår exakt samma kordinater och dimensioner för begränsningsområden som de i märkta data.

Intersection over union (IoU) är ett evalueringsmått som används för att lösa problemet med hjälp av att beräkna hur bra två begränsningsområden överlappar. Flera av identifieringsalgoritmerna använder detta mått för att utvärdera hur bra algoritmen presterar vid träningsskedet. För arean B_1 och B_2 av två begränsningsområden räcknas IoU på följande sätt.

$$IoU = \frac{B_1 \cap B_2}{B_1 \cup B_2} \tag{2}$$

Figur 4 visar hur begränsningsområden med högre IoU överlappar bättre än de med lägre IoU.

Det är vanligt för metoderna att använda ett på förhand tränat klassifierings nätverk som basis för arkitekturen. Föremålsidentifierings delen av modellen läggs ofta efter detta. På det här sättet används den första delen av nätverket för att extrahera särdrag som föremålsidentifieringen kan vidare använda.

4.1 Evaluering

För evaluering av hur en modell presterar används flera olika kriterier. De vanligaste kriterierna är noggrannhet och återkallelseförmåga. Nogrannhet mäter hur hur stor del av de förutspådda positiva värdena är korrekta och återkallelseförmåga hur bra modellen hittar alla positiva resultat. Matematiskt definieras dessa på följande sätt:

$$T_P = \text{Sant positiv} \tag{3}$$

$$F_P = \text{Falskt positiv} \tag{4}$$

$$F_N = \text{Falskt negativ} \tag{5}$$

$$\begin{aligned} P &= \frac{T_P}{T_P + F_P} \\ R &= \frac{T_P}{T_P + F_N} \end{aligned} \tag{6}$$

Där P är noggranhet(eng. precision) och R åtsekallelseförmåga(eng. recall).

Ett evalueringsmått som baserar sig på dessa är Average Precision(AP). Det här måttet räknas ut av de 11 bästa resultaten från modellen för ett visst föremål. Det här måttet beräknas skillt för varje kategori.

(skriv mera)

Mean AP(mAP) används ofta för ett mått för prestanda. mAP räknas som ett medelvärdet av AP värdena för varje klass. [8]

(skriv mera och bättre)

4.2 Region based CNN

Region based CNN(R-CNN) introducerades av Ross Girshick år 2014 och kan tänkas som en naturlig utvidgning på bildklassifiering med hjälp av delområdes förslag [9]. R-CNN var den första metoden som använde en två-stegs-metod för föremålsidentifiering med hjälp av djupinlärning [8]. Istället för att göra en uttömmande sökning på delområden, använder metoden selektiv sökning eller någon liknande algoritm, för att begränsa områden och ett faltningsnätverk för att hitta särdrag i dessa. [9]

Selektiv sökning använder segmentering av en bild för att ge förslag på delområden som möjligtvis kan innehålla ett objekt. Algoritmen genererar en hierarki från små till stora delområden som kan användas för vidare klassifiering. [10]

Bilden segmenteras först med hjälp av grafbaserad segmentering - en algoritm som delar upp bilden i flera delområden[11]. Sedan grupperas liknande bildområden ihop i flera iterationer tills bara en grupp återstår. Detta görs med hjälp av att beräkna ett likhetsmått mellan områden som ligger bredvid varandra. Likhetsmåttet $s(r_i, r_j)$ mellan två bredvid varandra liggande områden r_i och r_j definieras som en kombination av fyra olika likhetskriterier: färg, textur, storlek och form.

Figur 5: Selektiv sökning

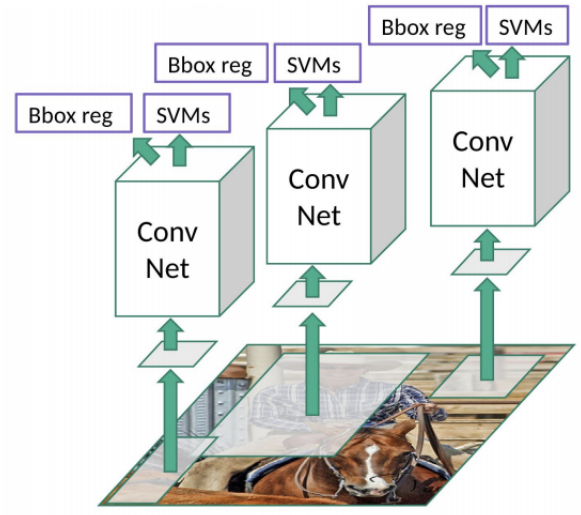
$$s(r_i, r_j) = s_{colour}(r_i, r_j) + s_{texture}(r_i, r_j) + s_{size}(r_i, r_j) + s_{fill}(r_i, r_j) \tag{7}$$

Områden med det största likhetsvärde för två bredvid varandra liggande områden kombineras till ett område. De kombinerade områdena i samband med de första segmenten kan användas som förslag på områden som innehåller objekt. 5 visualiserar iterationerna i en selektiv sökning.

RCNN börjar med att hitta delområdes förslag med hjälp av selektiv sökning. Selektiv sökning ger ungefär 2000 förslag på delområden av bilden som sedan vidare klassifieras. Det andra skedet använder ett faltningsnätverk för att hitta särdrag för varje delområde. Eftersom ett faltningsnätverk kräver att indata har samma storlek, förvrids delområderna till samma storlek före detta steg. Det tredje skedet är tvådelat, den ena delen klassifierar områderna med hjälp av SVM - en sorts klassifierare - som tränats för olika klasserna, medan den andra delen använder en linjär regressions modell för att göra korrigeringar på begränsningsregionerna. Orsaken för att inkludera regressions modellen är att selektiv sökning inte nödvändigtvis hittar exakt de rätta delområderna runt ett föremål. Vissa korrigeringar måste alltså göras. [9]

Vid träningsskedet tränas faltningsnätverket först med en datamängd menad för enkel klassifiering. Sedan finjusteras nätverket till att lokalisera och klassifiera förvridna delområdesförslag. Det sista lagret i det tränade nätverket ersätts med $(N+1)$ neuroner där N är antalet föremålersklasser. En ytterligare klass tillägs för bakgrund. Delområdesförslag med $IoU \geq 0.5$ ses som positiva klassifieringar och de övriga som negativa. [9]

(skriv mera)



Figur 6: De fyra stegen i R-CNN algoritmen.

4.3 Fast RCNN

En av de största nackdelen med RCNN när det gäller prestanda är att faltningsnätverket måste köras på varje skilld delområdesförslag. Fast RCNN är en förbättrning på RCNN algoritmen som strävar till att motverka det här. [12]

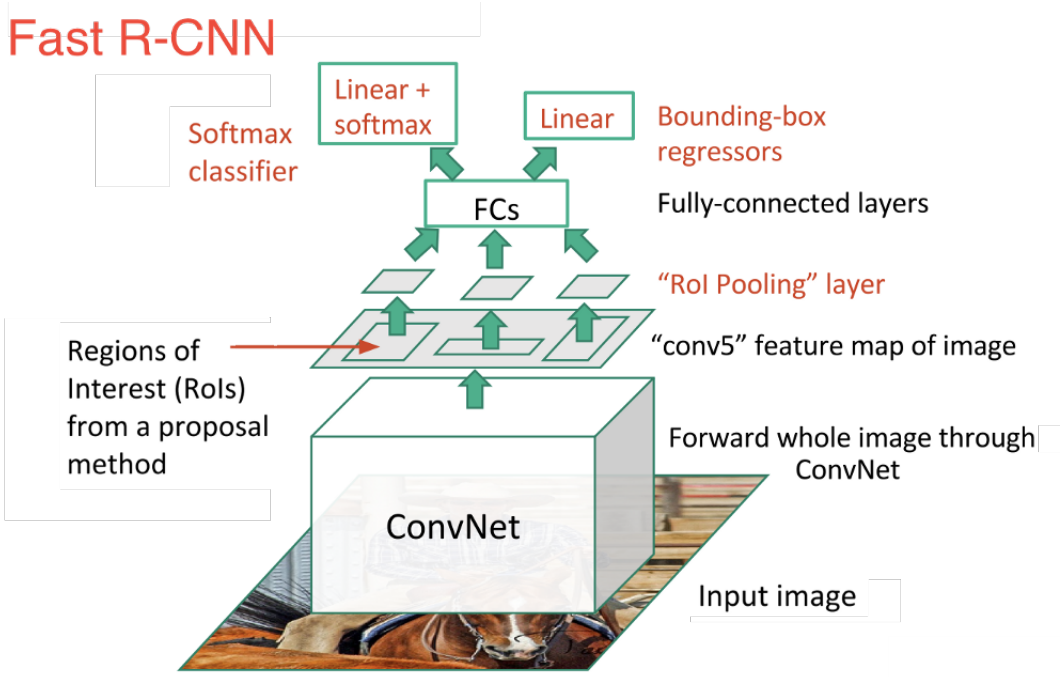
Fast RCNN är fortfarande en två-stegs-modell som RCNN, men istället för att först hitta delområden i en bild kör Fast RCNN genast hela bilden genom ett faltningsnätverk för att producera en aktiveringskarta. Delområdes förslag görs sedan på den här aktiveringskartan på motsvarande sätt som i RCNN. En så kallad RoI pooling lager(beskriv) används för att förvrida delområderna till samma dimensioner. [12]

Den största förbättringen med jämförelse till RCNN är i förminskning av beräkningstid. Noggranheten är fortfarande ungefär på samma nivå som RCNN. Träning av en Fast RCNN modell är också mycket snabbare på grund av att den kan göras på samma gång på regressionsdelen och klassifikationsdelen. Där det för RCNN tar ungefär 84 timmar att träna modellen tar det bara ungefär 9 timmar för Fast RCNN. [8]

Beräkningstiden för Fast RCNN domineras av tiden som tas åt för selektiv sökning. (skriv mera)

4.4 Faster RCNN

Faster RCNN är en förbättring på Fast RCNN introducerades år 2015. Faster RCNN motverkar Fast RCNN metodens begränsningar med hjälp av att istället för att använda t.ex. selektiv sökning, använda ett faltningsnätverk för delområdesförslag. Det här nätverket kallas för Region Proposal Network(RPN) och läggs genast efter aktiveringskartan som



Figur 7: Fast RCNN

genererats efter det första faltningslagren. Nätverket används för att förutspå begränsningsområden utan hänsyn till klass. Arkitekturen av Faster RCNN är visualiserad i figur 8. [13]

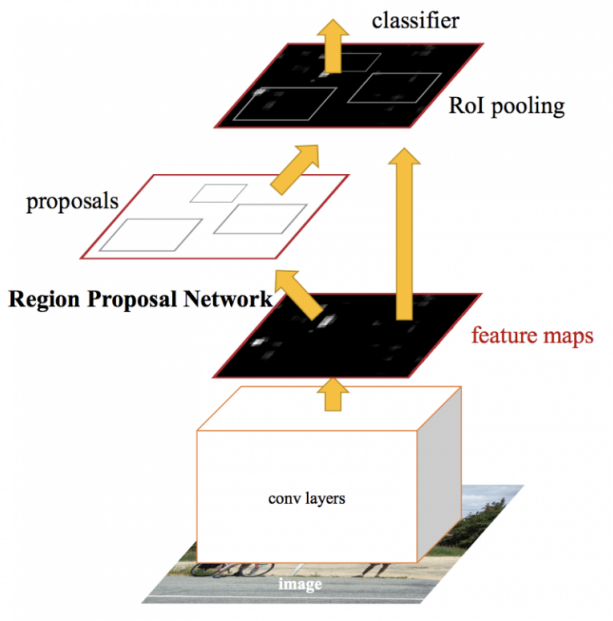
RPN använder som indata aktiveringskartan som fås som utdata från det sista faltningslagret. En nxn filter används med k områdesförslag i olika storleker som kallas ankare.

(skriv mera)

4.5 You Only Look Once

De tidigare metoderna har varit två-stegs-metoder där lokalisering av föremål görs med hjälp av att först söka delområden inom bilden och sedan klassifiera dessa. You Only Look Once (YOLO) är en metod som introducerades år 2016 av Joseph Redmon. YOLO är ett exempel på en ett-stegs-metod som baserar sig på ett enda faltningsnätverk som i ett och samma steg förutspår både områdesbegränsningar samt klassifierar dessa.

Metoden börjar med att dela upp bilden i ett rutnät. Varje ruta ansvarar för att förutspå ett bestämt antal begränsningsområden och ett värde på hur mycket fötroende modellen har att dessa områden innehåller föremål. 5 olika värden förutspås för områdena: x och y kordinater för mittpunkten av begränsningsområdet i relation till rutans kanter, längd och höjd på området och förtroendevärdet för att området innehåller ett föremål. Det här



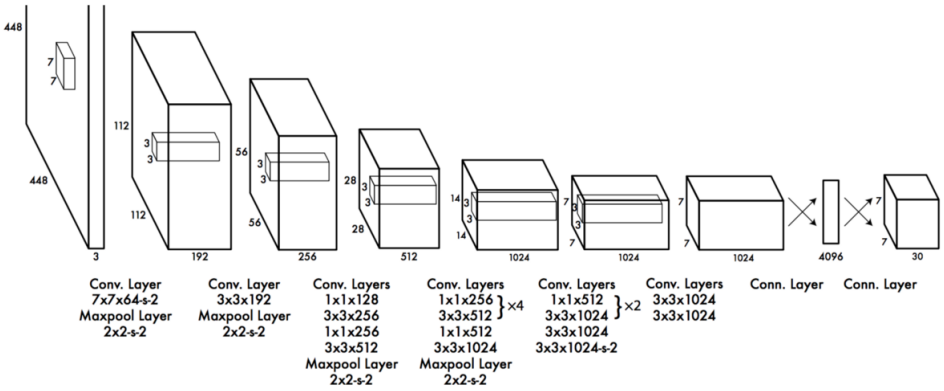
Figur 8: Faster RCNN arkitektur

förtroende värdet är ett mått på hur mycket förtroende modellen har i att rutan innehåller ett föremål samt hur nära rutans dimensioner är till de verkliga dimensionerna. Det här värdet definieras som $Pr(Objekt) * IOU$ där IOU är skillnaden mellan det förutspådda begränsningsområdet och det verkliga. Måttet säger inget om vilken klass föremålet tillhår, bara om det finns ett föremål eller inte och hur noggrann områdesbegränsningen är. Varje ruta i rutnätet ansvarar för att förutspå det objekt vars mittpunkt ligger innanför rutan. Eftersom varje cell i rutfältet innehåller flera förutsägelser för områdesbegränsningar görs en av dessa alltid ansvarig för att förutspå något visst föremål. Detta väljs på basen av den största IOU värdet med träningsdatan.

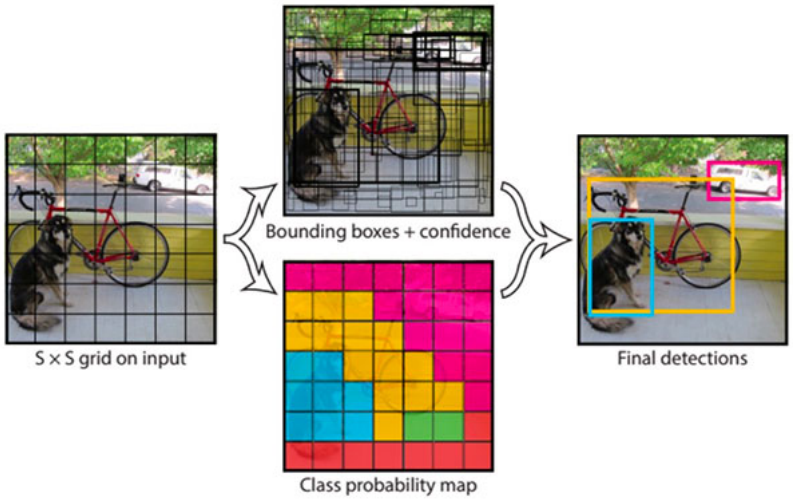
Förutom de här värdena förutspår också varje cell betingade sannolikhetsvärden $Pr(Klass_i|Objekt)$ för alla klasser. Det här är ett förtroende värde på att cellen innehåller ett föremål av klass i givet att cellen innehåller ett föremål. Bara ett sådant sannolikhetsvärde räknas ut för varje cell i rutnätet. Figur 10 visar hur bildens celler delas in i sannolikhetsvärden på basen av det största betingade sannolikhetsvärde för något föremål. De olika färgen representerar de olika klasserna. Förtroende värdet för ett begränsningsområde multipliceras med cellens betingade sannolikhets värde för att beskriva det slutliga värdet. [14]

$$Pr(Klass_i|Objekt) * Pr(Objekt) * IOU = Pr(Klass_i) * IOU \tag{8}$$

Arkitekturen på nätverket som beskrivs i artikeln har 24 faltningslager, 4 max-pooling lager och 2 fullt kopplade lager. Vart annat faltningslager har filter med $1x1$ storlek. Dessa används för att reducera utdatans djup från lagret. Det sista lagret innehåller information



Figur 9: YOLO arkitektur



Figur 10: YOLO rutfält

om sannolikheter för de förutspådda klasserna samt begränsningsområden till föremålet. Som i de tidigare metoderna förhandtränas nätverket först på enkel bildklassifierings data och därefter specialiseras till lokalisering och klassifiering. [14]

(skriv mera YOLOv2/YOLOv3)

5 Jämförelse av olika föremålsidentifieringsmetoder

En uppenbar jämförelse som kan göras är mellan två stegs och ett stegs metoder. Metoderna baserade på RCNN är alla exempel på två stegs metoder. Dessa har i allmänhet haft bättre framgång i noggrannhet medan ett stegs metoder som YOLO åstadkommit snabbare resultat med sämre noggrannhet. Eftersom Faster RCNN är en förbättring på Fast RCNN som är en förbättring på RCNN är det naturligt att Faster RCNN presterar bäst

av dessa. Förbättringarna som gjorts för Faster RCNN är betydelsefulla i att både öka noggranhet och hastighet, men metoden kan inte fungera i realtid [14].

Metod	mAP	FPS
YOLO	63.4	45
Fast RCNN	70.0	0.5
Faster RCNN VGG-16	73.2	7
Faster RCNN ZF	62.1	18

Tabellen visar en jämförelse mellan olika metoders mAP och FPS på PASCAL VOC 2007. Där varianterna av Faster RCNN har en bättre mAP ligger de betydligt efter i FPS.

(skriv mera)

6 Diskussion och sammanfattning

I avhandlingen beskrivdes och jämfördes moderna metoder för föremålsidentifiering med hjälp av djupinlärning. Framstegen som gjorts i bara några år är lovande. Lokalisering och klassifiering av flera föremål i en bild som tidigare tagit flera sekunder kan redan i dagens läge göras med hög precision i realtid.

Trots de stora framstegen inom området finns det ännu utmaningar. Ett stort problem med djupinlärningsmodeller i dagens läge är deras kännslighet till data som modellen inte var byggt för att hantera. Då flera tillämpningar av modellerna tas i bruk inom system som kräver en viss robusthet och säkerthetsgaranti blir frågan av modellernas pålitlighet aktuellt. Djupinlärningsmodeller har visat sig vara enkla att manipulera med konstgjord data. Med vissa små ändringar i indata kan man få algoritmen att ge felaktiga svar. Man kan även göra manipulationerna i indata på ett sådant sätt där en människa inte märker skillnaden. [15]

(skriv mera)

Källförteckning

[1] A. Samuel, ”Some studies in Machine Learning using the game of checkers”, *IBM Journal of Research and Development*, årg. 3, juli 1959.

[2] I. Goodfellow, Y. Bengio och A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[3] S. Ruder, ”An overview of gradient descent optimization algorithms”, *CoRR*, årg. abs/1609.04747, 2016. arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[4] *CS231n Convolutional Neural Networks for Visual Recognition*, http://cs231n.github.io, Läst: 3.3.2019.

[5] *Neural Networks and Deep Learning*, http://neuralnetworksanddeeplearning.com, Läst: 14.3.2019.

[6] Z. Zhao, P. Zheng, S. Xu och X. Wu, ”Object Detection with Deep Learning: A Review”, *CoRR*, årg. abs/1807.05511, 2018. arXiv: 1807.05511. URL: http://arxiv.org/abs/1807.05511.

[7] K. S. Chahal och K. Dey, ”A Survey of Modern Object Detection Literature using Deep Learning”, *CoRR*, årg. abs/1808.07256, 2018. arXiv: 1808.07256. URL: http://arxiv.org/abs/1808.07256.

[8] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu och M. Pietikäinen, *Deep Learning for Generic Object Detection: A Survey*, 2018. eprint: arXiv:1809.02165.

[9] R. B. Girshick, J. Donahue, T. Darrell och J. Malik, ”Rich feature hierarchies for accurate object detection and semantic segmentation”, *CoRR*, årg. abs/1311.2524, 2013. arXiv: 1311.2524. URL: http://arxiv.org/abs/1311.2524.

[10] J. Uijlings, K. van de Sande, T. Gevers och A. Smeulders, ”Selective Search for Object Recognition”, *International Journal of Computer Vision*, 2013. DOI: 10.1007/s11263-013-0620-5. URL: http://www.huppelen.nl/publications/selectiveSearchDraft.pdf.

[11] P. F. Felzenszwalb och D. P. Huttenlocher, ”Efficient Graph-Based Image Segmentation”, *Int. J. Comput. Vision*, årg. 59, nr 2, s. 167–181, sept. 2004, ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000022288.19776.77. URL: https://doi.org/10.1023/B:VISI.0000022288.19776.77.

[12] R. B. Girshick, ”Fast R-CNN”, *CoRR*, årg. abs/1504.08083, 2015. arXiv: 1504.08083. URL: http://arxiv.org/abs/1504.08083.

[13] S. Ren, K. He, R. B. Girshick och J. Sun, ”Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, *CoRR*, årg. abs/1506.01497, 2015. arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[14] J. Redmon, S. K. Divvala, R. B. Girshick och A. Farhadi, ”You Only Look Once: Unified, Real-Time Object Detection”, *CoRR*, årg. abs/1506.02640, 2015. arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras och A. Vladu, ”Towards Deep Learning Models Resistant to Adversarial Attacks”, *CoRR*, årg. abs/1706.06083, 2017. arXiv: 1706.06083. URL: http://arxiv.org/abs/1706.06083.