Djupinlärning

Faltningsnätverk för styrsystem i självkörande bilar

Utkast 10.4.2018

Sebastian Tallberg

REFERAT

1. Inledning

TODO

2. Bakgrund

2.1 Maskininlärning

Begreppet maskininlärning introducerades år 1959 av Arthur Samuel, känd för sin forskning inom artificiell intelligens [1]. Samuel gav följande beskrivning: "Maskininlärning är ett ämnesområde som ger datorer kunskapen att lära sig utan att vara explicit programmerade" [1]. Maskininlärningsalgoritmer ger möjligheten att hantera problem som annars vore svåra, om inte omöjliga, att lösa med traditionella programmeringsmetoder [3]. Några exempel på sådana algoritmer är linjär- och logistisk regression, k-närmaste grannar, beslutsträd, stödvektormaskin och neurala nätverk. Tillämpningsområden för system som bygger på maskininlärning är bland annat bild- och objektigenkänning, taligenkänning, sökmotoroptimering [2], spamfiltrering, robotteknik, hälsovård och e-handel [1].

Många av de traditionella maskininlärningsalgoritmerna är dock begränsade i hur de extraherar information från data [2]. Det kräver ofta väsentlig domän-expertis och noggrann planering för att förvandla råa data till en meningsfull representation för det egentliga maskininlärningssystemet [2]. Detta kallas för feature engineering(?). Då man förflyttar sig till dagens situation förekommer ett nytt delområde av maskininlärning, djupinlärning, som har strävat efter att lösa detta problem.

2.2 Djupinlärning

Djupinlärning (eng. deep learning) är en gren av maskininlärning som fokuserar på metoder som använder sig av djupa artificiella neurala nätverk [3]. Dessa algoritmer baserar sig på inlärning av särdrag (feature learning), vilket innebär att nätverket är bra på att lära sig de behövliga representationerna direkt från råa inputdata [2]. Data transformeras till en representation, som sedan transformeras till en mer abstrakt representation, och så vidare, beroende på antalet lager i neurala nätverket [2]. Detta är en av de främsta fördelarna med djupinlärning, eftersom det

ger möjligheten att konstruera komplicerade algoritmer som tar fasta på de viktiga detaljerna och exkluderar det oväsentliga [2].

I dagens läge används ofta begreppet djupinlärning och artificiella neurala nätverk synonymt. Det är dock viktigt att påpeka att utvecklingen av ANN daterar sig tillbaka ända till 1950- och 1960-talet, då Frank Rosenblatt uppfann perceptronen; en typ av artificiell neuron som tar in flera binära input och ger ut ett binärt output [9]. Neurala nätverken har genomgått flera förbättringar och arkitekturella förändringar sen dess, och idag har djupa neurala nätverk blivit en populär typ av maskininlärningsalgoritm. Utvecklade algoritmer för effektiv träning, kraftfullare datorer samt lättare åtkomst till stora mängder data är bland de orsakade faktorerna.

3. Artificiella neurala nätverk

Ett artificiellt neuralt nätverk (ANN) är en matematisk modell, eller algoritm, som är inspirerad av det biologiska neurala nätverket hos människan [4]. Strukturen för ett neuralt nätverk består av flera sammankopplade artificiella neuroner, som är grupperade i olika så kallade lager. Den simplaste typen av nätverk har åtminstone ett inputlager och ett outputlager. De mer sofistikerade djupa nätverken som används idag består också av flera dolda lager mellan input- och outputlagret, vilket ger möjligheten att lösa mer komplexa problem [9]. Målet med ett neuralt nätverk är att försöka uppskatta en funktion $y = f(x; \theta)$, som tar in någon input x och ger ut ett output y, genom att hitta den underliggande strukturen i data som matas in vid inlärning [3][10]. Inlärningsprocessen går ut på att hitta de rätta parametrarna θ som bidrar till det bästa resultatet [3].

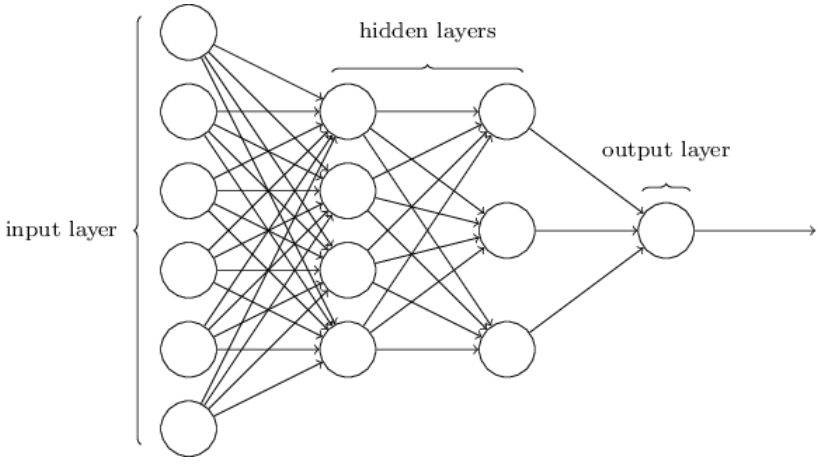
Arkitekturen hos ett ANN bestämmer dess beteende; hur nätverket lär sig och vad det är som den lär sig. Till detta hör bland annat hur djupt nätverket är, hur många neuroner det finns i de olika lagren, samt hurdan kopplingen är mellan neuronerna [4]. Dessa kallas för hyperparametrar; parametrar i nätverket som bestäms då man

planerar arkitekturen. Den typen av ANN som kommer att presenteras till följande är ett standard framkopplingsnätverk. Faltningsnätverk som introduceras i kapitel 4 är en utvidgning av denna variant.

3.1 Framkopplingsnätverk

Framkopplingsnätverk är den vanligaste typen av neurala nätverk, där informationsflödet sker endast framåt. Detta innebär att outputen från neuronerna i ett lager fungerar som inputen till neuronerna i det följande lagret [9]. Den enklaste varianten av dessa nätverk är fullt anslutna, vilket betyder att varje enskild neuron är kopplad med alla andra neuroner i de intilliggande lagren [4]. Figur x visar strukturen för denna typ av nätverk.

Första lagret i nätverket är inputlagret, som består av ett visst antal inputneuroner beroende på hurdana data som matas in vid inlärningen. Sista lagret, eller outputlagret, består av en eller flera outputneuroner, också beroende av inputdata [9]. Regression, där outputen är ett kontinuerligt värde, eller binär klassificering, där outputen är någon sannolikhet, kan åstadkommas med endast en outputneuron. Vid klassificering av exempelvis bilder i olika kategorier, så motsvarar antalet output neuroner de möjliga kategorierna. Mellan inputlagret och outputlagret finns de dolda lagren, vars noder kallas för dolda neuroner. Ett nätverk med två eller flera dolda lager brukar definieras som djupt [4].

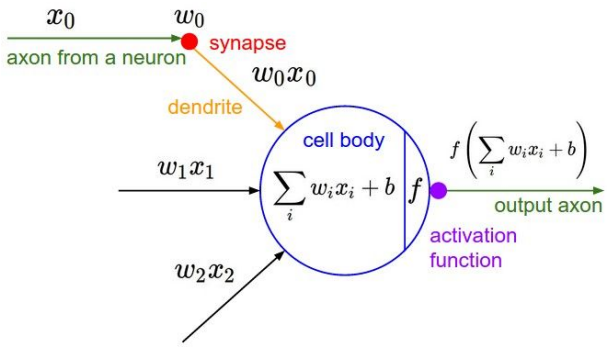


Figur x: Fullt anslutet framkopplingsnätverk [9]

3.2 Artificiella neuronen

Den centrala delen av ett neuralt nätverk är den artificiella neuronen. Neuronerna i ett neuralt nätverk får någon input, som sedan transformeras till en output med hjälp av en icke-linjär aktiveringsfunktion. Undantaget är neuronerna i inputlagret, som får sin input från utsidan av nätverket. Dessa neuroner skickar värdet vidare till det första dolda lagret utan någon slags aktivering.

I ett fullt anslutet framkopplingsnätverk skickas neuronens output vidare som input till alla andra neuroner i det följande lagret [4]. Varje koppling mellan neuronerna i nätverket har ett skilt värde, en vikt, som beskriver hur viktig den kopplingen är. Dessutom har varje neuron i de dolda lagren samt outputlagret också en bias(sve?) term *b*, som ytterligare påverkar neuronens output. Inputen till en neuron är summan av de viktade värden från förra lagret, samt neuronens bias värde *b*, som sedan körs igenom aktiveringsfunktionen för att producera outputen. Formeln för detta är $\sigma(b + \sum_{i=1}^{n} w_i x_i)$, där σ är aktiveringsfunktionen. Figur x visualiserar detta steg.

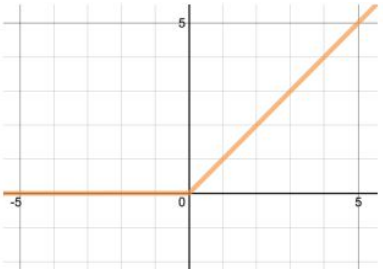


Figur x: En artificiell neuron [11]

Aktiveringsfunktionens uppgift är att implementera icke-linjäritet i modellen. Den ska vara en deriverbar funktion, eftersom backpropagation algoritmen som används för att justera parametrarna i nätverket går ut på att beräkna gradienter.

Sigmoid-funktionen f(x) = $\frac{1}{1+e^{-x}}$ är ett exempel på en aktiveringsfunktion som tar en input och producerar en output mellan 0 och 1, och var länge den populäraste

aktiveringsfunktionen. Idag har ReLU-aktiveringar (rectified linear unit) tagit över, eftersom de har visat bättre prestanda i olika situationer. De drabbas heller inte av försvinnande gradienter på samma sätt som sigmoid-funktionen [4].
ReLU-aktiveringen definieras som f(x) = max(0, x), där outputen alltid är 0 eller större. Då inputen är högre än tröskelvärdet 0 förekommer det ett linjärt samband mellan inputen och outputen.



Figur x: ReLU-aktivering

3.3 Förlust funktion

Då man tränar ett neuralt nätverk är man intresserad av att hitta de rätta parametrarna, vikter och biases(?), som producerar den bästa möjliga utmatning. För att åstadkomma detta används något som kallas för en förlustfunktion, också kallad en kostfunktion. Förlustfunktionens syfte är att meddela hur nätverket presterar under inlärningsprocessen. Ju lägre funktionens output är, desto bättre resultat producerar nätverket [9].

Det förekommer ett flertal förlustfunktioner, varav en av de enklaste är MSE (mean squared error), som beräknar medelvärdet av det kvadratiska felet mellan nätverkets output för alla input x_i och de förväntade outputen y_i. Formeln ser ut på följande vis:

$$MSE = \frac{1}{m} * \sum_{i=1}^{m} (h(x_i) - y_i)^2$$

Vilken förlustfunktion som används beror på hurdana data man har och vad det är man vill lära nätverket. MSE är exempelvis passande för regression, där outputen är

ett kontinuerligt värde. Syftet med varje förlustfunktion är dock alltid densamma; visa hur bra nätverket presterar. Förlustfunktionen ska på samma sätt som aktiveringsfunktionen också vara deriverbar, eftersom backpropagation algoritmen måste kunna beräkna gradienten för att optimera parametrarna.

3.4 Backpropagation

Backpropagation is a widely used learning algorithm for artificial neural networks in supervised learning, where the expected output of the training data is known. The goal is to minimize the error of the model on the training data, where the error is given by the chosen loss function. If the output of the model matches the expected output, nothing is done. However, if an error does exist, the parameters of the network, i.e. the weights and biases, are adjusted to minimize the error [5]. To know how to adjust these parameters, the gradient of the loss function is calculated. The name backpropagation comes from the fact that the error is calculated at the output and then distributed backwards through the hidden layers.

-chain rule, gradient descent

TODO: översätt, skriv om, visa matematiskt

3.5 Inlärningsmetoder

Sättet att träna en modell kan indelas i olika metoder baserat på hurdana data som matas in i modellen, samt problemet man försöker lösa. De två vanligaste inlärningsmetoderna är övervakad- och oövervakad inlärning.

3.5.1 Övervakad inlärning

Den mest framgångsrika formen av maskininlärning för tillfället är övervakad inlärning. Övervakad inlärning beskriver metoden då modellen tränas på en

datamängd som har en färdigt given output för varje input. Eftersom outputen är känd kan modellen justera parametrarna, eller vikterna, enligt felet. Övervakad inlärning används i största delen av praktiska tillämpningar av maskininlärning [3]. I detta arbete kommer tyngdpunkten att ligga på denna form av inlärning.

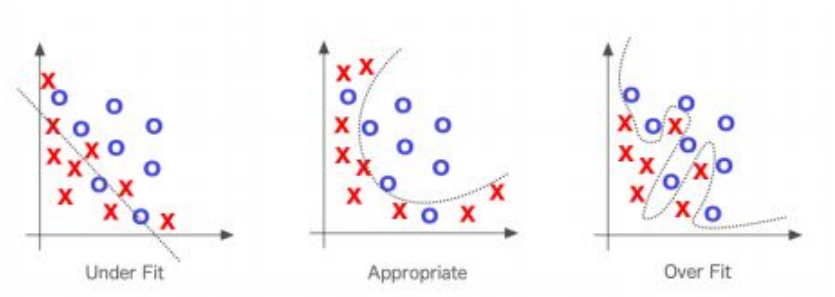
3.5.2 Oövervakad inlärning

På andra sidan av spektrumet förekommer oövervakad inlärning, där de data som används för inlärning saknar den korrekta outputten. I denna metod hittar algoritmen något mönster i data, men det finns inget rätt eller fel svar. Klusteranalys, där data grupperas i olika kluster baserat på likheter, är ett exempel på oövervakad inlärning [3]. Medan övervakad inlärning är för tillfället den dominerande formen av maskininlärning, så anser Yann LeCun et al. [3] att oövervakad inlärning kommer att spela en viktigare roll på längre sikt.

3.6 Över- och underanpassning

En framgångsrik implementation av en modell betyder att datorn har lärt sig den underliggande strukturen av träningsdata. Då modellen har lärt sig den underliggande strukturen, och inte bara själva strukturen, kan den prestera bra i verkliga situationer. Fenomenet då modellen kan göra förutsägelser med mycket bra noggrannhet på träningsdata, men presterar dåligt i verkligheten, kallas för överanpassning. Orsaker till detta kan vara att datamängden är för liten, eller den använda modellen är för komplex jämfört med det tillgängliga data. Detta problem löses med hjälp av reglering, anskaffning av mera data, eller att försöka använda en annan arkitekturmodell [7].

En underanpassad modell beskriver situationen då den underliggande strukturen kan ej fångas. Detta problem kan lösas med att introducera flera dolda lager i nätverket, ändra på antalet neuronen per lager, eller att försöka använda en annan arkitektur [7].

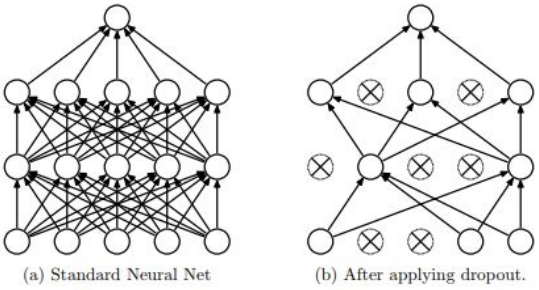


Figur x: Över- och underanpassning [4]

3.6.1 Reglering

Ett sätt att strida emot överanpassning kallas för reglering (eng. regularization). Goodfellow et. al [3] beskriver det på följande vis: “Reglering är någon ändring vi gör till en inlärningsalgoritm som är avsedd för att minska dess generaliseringsfel, men inte dess träningsfel”. Det har visar sig att det inte finns någon regleringsmetod som är bäst för alla problem, utan den optimala metoden beror på problemet som löses samt vilken typ av inlärningsalgoritm som används. Reglering tillhör de viktigaste problem inom maskininlärning. En populär regleringsmetod för neurala nätverk är dropout [3].

Dropout är en kraftfull samt beräkningsmässigt billig regleringsmetod, och har visat sig vara passande för djupa neurala nätverk [3][8]. Med dropout stänger man slumpmässigt av ett visst antal neuroner samt deras kopplingar i de olika lagrena under träningen [4][8]. Detta demonstreras i figur x.



Figur x: Dropout [8]

4. Faltningsnätverk

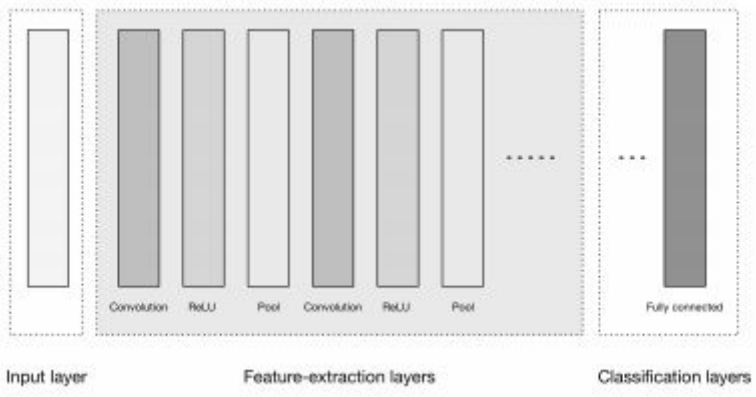
Faltningsnätverk är en klass av djupa framkopplingsnätverk som har visat stor framgång inom datorseende, men också för naturlig språkbehandling och ljudanalys [4]. De är Inspirerade av människan syncentrum, och är bra på att lära sig olika särdrag i data genom en process som kallas för faltning [4]. De introducerades år 1989 av Yann LeCun, och var bland de första nätverksarkitekturerna som hittade kommersiell framgång, och används idag i stor utsträckning [4]. År 2012 vann Alex Krizhevsky [12] den årliga ImageNet klassificeringstävlingen med ett djupt faltningsnätverk, som presterade betydligt bättre än tidigare lösningar. Detta visade styrkan hos faltningsnätverk för bildigenkänning, och är också en av de ledande orsakerna bakom det stora intresset för djupinlärning idag [3]. Detta kapitel presenterar en översikt av arkitekturen och de specifika egenskaperna som gör faltningsnätverk effektiva för att hantera bilder som input.

4.1 Arkitektur

Faltningsnätverkarkitekturen är en klass av framkopplingsnätverk, som presenterades i kapitel 3.1. Problemet med traditionella fullt anslutna framkopplingsnätverk är att de klarar inte av att effektivt hantera högdimensionell data som exempelvis bilder [4]. Anta ett nätverk som klassificerar färgbilder med storlek 300x300. Detta skulle leda till 300*300*3(RGB-värdena) = 270 000 vikter och anslutningar för varje neuron i det första dolda lagret. I ett djupt nätverk vill man ytterligare ha flera av dessa lager, vilket snabbt leder till många parametrar i nätverket. Förutom att det leder till en längre inlärningstid för nätverket, så är överanpassning också ett potentiellt problem [11]. Faltningsnätverk ger en bättre lösning på detta problem.

Faltningsnätverk är strukturerade att ta nytta av bildernas tredimensionella struktur, vilket innebär att neuronerna är ordnade i tre olika dimensioner: bredd,

höjd samt djup [11]. Inputlagret tar in en tredimensionell inputvolym, en bild, som sedan skickas vidare till de lagren som utför särdragsextraktion. Denna del av nätverket är en kombination av faltningslager, ReLU-aktiveringar samt pooling-lager, som tillsammans hittar de viktiga särdragen i data. Sista delen av ett faltningsnätverk består av ett eller flera fullt anslutna lager, som beräknar det slutliga output värdet beroende på om det är klassificering eller regression [4]. Figur x ger en översikt av ett typiskt faltningsnätverk.

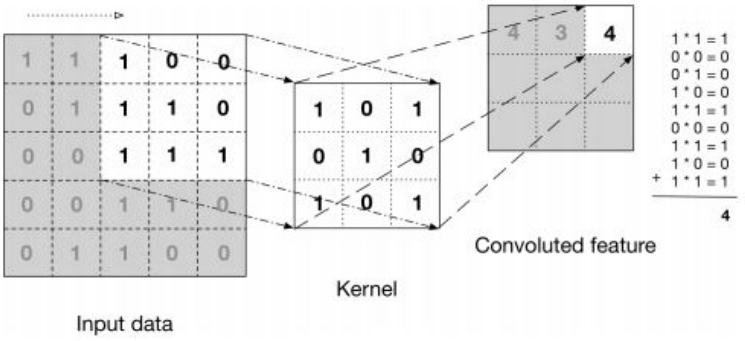


Figur x: översikt av ett faltningsnätverk [4]

4.2 Faltningslager

Den viktigaste delen av ett faltningsnätverk är själva faltningslagren. Faltningslagrens uppgift är att ta en tredimensionell input och producera en tredimensionell output i form av särdragskartor. Detta görs med hjälp av så kallade filter, som dras över inputvolymen. Varje filter har en storlek FxF (bredd och höjd), samt ett kliv S (eng. stride). Klivet anger hur många pixlar filtret flyttar sig på inputvolymen. Djupet på ett filter motsvarar alltid inputvolymens djup [4][11].

För varje position som filtret kan läggas över inputvolymen räknar man skalärprodukten, som visas i figur x, och outputen blir en tvådimensionell särdragskarta. Bredden och höjden på den nya outputen beror på filtrets storlek samt kliv, eftersom dessa egenskaper påverkar antalet positioner i inputvolymen som filtret kan appliceras. Varje faltningslager består vanligtvis av många filter (varje filter har sammas storlek och kliv), och eftersom varje filter producerar exakt en särdragskarta, innebär det att djupet på faltningslagrets output är samma som antalet filter i lagret, eller antalet särdragskartor [4][11]. Dessutom utför man elementvis ReLU-aktivering på varje särdragskarta efter ett faltningslager, som introducerar icke-linjäritet i modellen.



Figur x: faltning [4]

Tre distinkta och kraftfulla egenskaper av ett faltningsnätverk är delning av vikter, fåtaliga anslutningar mellan neuroner samt translation invariance(sve?). Anta exempelvis ett 2x2x3 filter som dras över en 5x5x3 input med ett kliv på 1. Detta producerar en 4x4 särdragskarta. Varje filter har 2*2*3 + 1(bias) = 13 parametrar, och dessa används för att beräkna varje värde i den nya 4x4 särdragskartan. Motivationen bakom detta är att ett filter som har lärt sig att upptäcka något särdrag som exempelvis kanter, kommer troligen att vara användbar över hela bilden och inte bara en del av den. Delade vikter betyder också att faltningsnätverk är translation invariant; de kan upptäcka objekt i en bild oberoende av dess position. Faltningslager leder också till glesa förbindelser där varje output i ett lager är endast beroende av några input, i motsats till ett fullt anslutet nätverk. Tillsammans bidrar dessa saker till färre parametrar, djupare nätverk, snabbare inlärning samt mindre sannolikhet att överanpassa modellen [3][4][11].

4.3 Pooling-lager

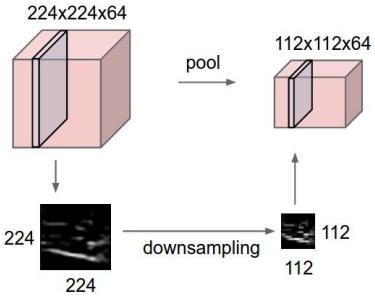
Medan faltningslagren är den centrala delen av ett faltningsnätverk, så implementerar många nätverk också ett eller flera pooling-lager. Syftet med ett pooling-lager är att krympa storleken på särdragskartorna (nedsampling) från det föregående faltningslagret. Därefter reduceras antalet parametrar i nätverket, som också hjälper till att förhindra överanpassning. Pooling operationen görs enskilt på varje särdragskarta, som innebär att djupet på inputen inte förändras, endast bredden och höjden [11][4]. Detta visas i figur x.

Pooling operationen utförs med ett filter som dras över inputen och skapar en ny output, på liknande sätt som i faltningslagret. Pooling-lagret har inga parametrar som justeras under inlärningen, men har dock 2 hyperparametrar som dikterar storleken på outputen; storleken på filtret F och dess kliv S (stride) [11]. Pooling-lagret får en input med storleken W_{in} X H_{in} X D_{in} och producerar en output W_{out} X H_{out} X D_{out}, där

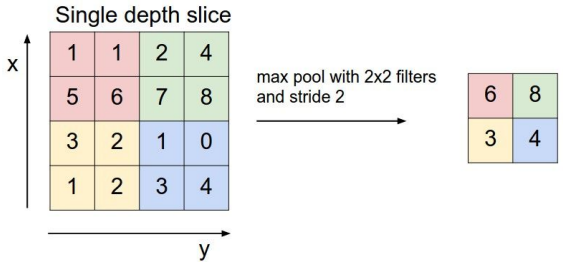
$$W_{out} = \frac{W_{in} - F}{S+1} \qquad H_{out} = \frac{H_{in} - F}{S+1} \qquad D_{out} = D_{in}$$

Den vanligaste pooling operationen är max-pooling, som visas i figur x. För varje FxF stort delområde som filtret glider över tas det största värdet, och en ny särdragskarta skapas som output. Det förekommer också bland annat average-pooling och L2-norm pooling, men max pooling har visar sig vara den mest använda och framgångsrika metoden [11].

Det finns också en åsikt att pooling lagren bör förkastas helt och hållet i faltningsnätverken. I artikeln "Striving for Simplicity: The All Convolutional Net" [13] föreslår skribenten att man kan ersätta pooling lagren med faltningslager, vars filter har ett högre kliv, och åstadkomma samma resultat i flera olika tester.



Figur x: Nedsampling av input [11]



Figur x: Max-pooling beräkning för enskild särdragskarta [11]

4.3 Fullt anslutna lager

TODO

5. PilotNet

I detta kapitel presenteras PilotNet-systemet, tidigare känt som DAVE-2, som är utvecklat av NVIDIA. Det är ett styrsystem för självkörande bilar som är uppbyggt av ett enda faltningsnätverk. Systemet tar in bilder från en framåtvänd kamera bakom vindrutan som input, och ger ut styrvinkeln för ratten som output. PilotNet utnyttjar faltningsnätverkets förmåga att bearbeta visuell data och ett koncept som kallas för end-to-end inlärning. End-to-end inlärning i detta sammanhang innebär att systemet lär sig att styra en bil genom att observera mänskliga förare, med endast videodata ihopkopplad med rätt styrvinkel som träningsdata. Nätverket är inte tränat för att exempelvis upptäcka bilar på vägen, fotgängare, körfältsmarkeringar eller annat. I stället lär sig nätverket de nödvändiga representationerna automatiskt. Detta är ett mycket annorlunda tillvägagångssätt för att utveckla autonom körning i bilar jämfört med andra metoder, som går ut på att bryta ner problemet i mindre delar.

Det nuvarande PilotNet-systemet är fortfarande ett pågående projekt och kan för tillfället bara styra bilen inom ett körfält. Utvecklarna bakom systemet tror dock att denna end-to-end lösning kommer att i slutändan leda till bättre prestanda och mindre system. Detta beror på att systemet har färre bearbetningssteg och optimerar dem samtidigt. Följande text kommer att presentera träningsprocessen och den nätverksarkitektur som används. Slutligen visas en visualisering av vilka delar av input bilden som dikterar vilken styrvinkel nätverket ger som output.

5.1 Inlärning

Todo: översätt och mera info

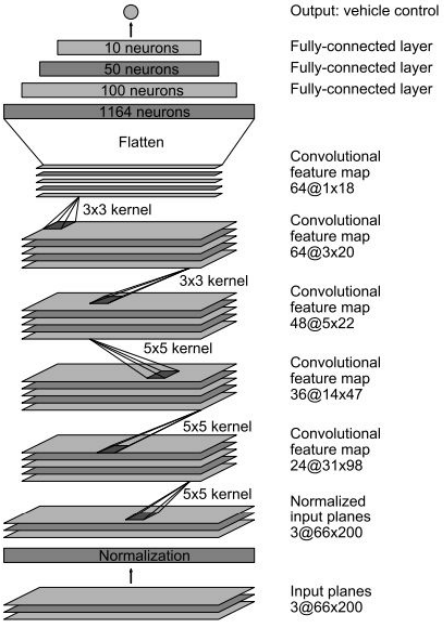
The training data used for learning the network has been gathered by humans driving a data collection car. The evaluated system was trained on under 100 hours

of driving footage that was gathered in different weather conditions and road types (både med och utan vägmarkeringar) during both day and night. Since the system is trained only to do lane following, only the data where the driver was staying in a lane is used during training. This video data is then sampled at 10 fps, as a higher sampling rate only resulted in highly similar images. When training the network, the input is an image taken from the video feed coupled with the steering command at that point.

While the trained network can output steering commands from only a single front facing camera, the training is done using three cameras behind the windshield; one in the center, one to the left and one to the right. The left and right cameras are used to create augmented data, so that the system can also learn to recover from mistakes. This augmented data shows the car in different positions and rotations from the road. The steering command for the augmented images is adjusted to one that would bring the car back to the wanted position in two seconds.

5.2 Nätverksarkitektur

PilotNet systemet består av ett enda faltningsnätverk med 9 olika lager, som demonstreras i figur x. Nätverket tar som input en färgbild med storlek 66x200 (höjd och bredd) som först skickas till ett normaliseringslager, vars uppgift är att normalisera pixelvärdena. Detta lager är hårdkodat och har därför inga parametrar som justeras vid inlärning. De tre första faltningslagren består av 5x5 stora filter med ett kliv på 2, medan de två sista faltningslagren använder 3x3 filter med ett kliv på 1. Dessa hyperparametrar valdes empiriskt genom att utföra tester med olika konfigurationer. Outputen från det sista faltningslagret plattas ut och skickas sedan vidare till 3 stycken fullt anslutna lager som producerar det slutliga styrkommandot.



5.3 Visualisering av framträdande objekt

TODO:

-Framträdande objekt, vad anser nätverket vara viktigt i input bilden

-VisualBackProp för att visualisera

-Resultat: nätverket tar i beaktande de viktiga delarna

6. Diskussion

Todo:

KÄLLOR

TODO: ska fixas

[1] Machine learning hands on for developers – Jason Bell (sid 2-3)

[2] Deep learning - Yann LeCun

https://www.researchgate.net/publication/277411157_Deep_Learning

[3] Deep learning - Ian Goodfellow et.al

http://www.deeplearningbook.org

[4] Deep learning - A practitioners approach (O'Reilly)

[5] Artificial Intelligence - A modern approach 3d edition

[6] Neural networks for pattern recognition - Christopher Bishop

[7] https://legacy.gitbook.com/book/leonardoaraujosantos/artificial-inteligence/details

[8] Dropout: A simple way to prevent neural networks from overfitting

http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

[9] http://neuralnetworksanddeeplearning.com

[10] Function Approximation Using Artificial Neural Networks

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.331.6019&rep=rep1&type=pdf

[11] http://cs231n.github.io/convolutional-networks/

[12] Visualizing and understanding convolutional neural networks

https://arxiv.org/pdf/1311.2901.pdf

[13] Striving for Simplicity: The All Convolutional Net

https://arxiv.org/pdf/1412.6806.pdf

[14] ImageNet classification with deep convolutional neural networks

http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[15] End to End learning for self-driving cars

https://arxiv.org/pdf/1604.07316v1.pdf

[16] Explaining how a deep neural network trained with end-to-end learning steers a car

https://arxiv.org/pdf/1704.07911.pdf

[17] VisualBackProp

https://arxiv.org/pdf/1611.05418.pdf