Dyp læring

Sigmund Rolfsjord

Oversikt

1. Grunnleggende om dyp læring og nevrale nett
2. Konvolusjonsnett
3. Synsfelt med konvolusjonsnett
4. Lurt å tenke på
5. Noen bruksområder

Lær mer:

Kurs fra Stanford: http://cs231n.stanford.edu/

Mer inngående bok:
http://www.deeplearningbook.org/

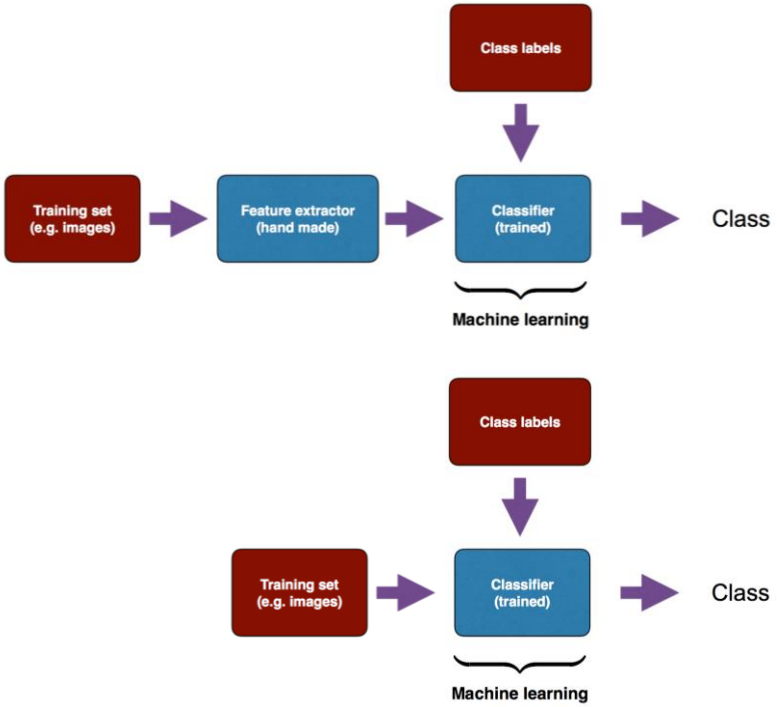
Programmerings verktøy:

- Pytorch
- Tensorflow

Dyp læring

En dominerende rolle i maskinsyn:

- Gode resultater
- Løser vanskelige problemer
  - Klassifisering
  - Segmentering
- Fleksibelt
- Kan lære direkte fra piksler



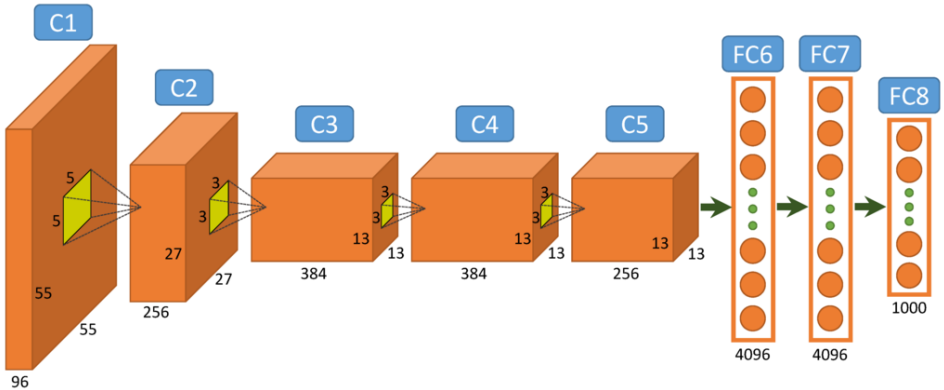
Dyp læring

$$f = Wx$$

$$f = W_2 \max(0, W_1 x)$$

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

- Hovedmål å tilnærme funksjon
- Kjennetegnes ved mange lag
- **Enkel form:** Matrisemultiplikasjon og ikke-lineær funksjon
- Ofte brukt sammen med konvolusjoner



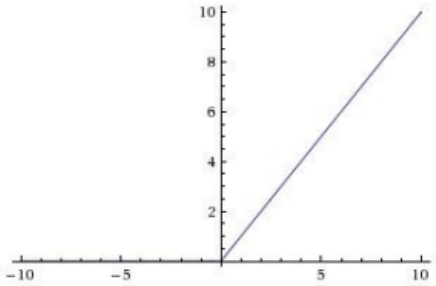
Dyp læring

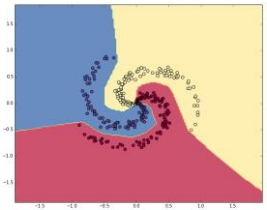
$$f = Wx$$

$$f = W_2 \max(0, W_1 x)$$

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

- De ikke-lineære funksjonene er viktige
- Her max(0, x), ofte kalt ReLu
  - Rectified Linear Unit
- Viktige egenskaper:
  - Gode gradienter
  - Effektive

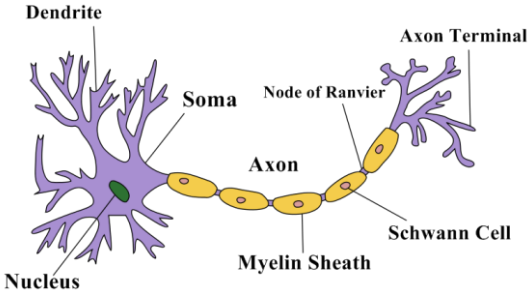


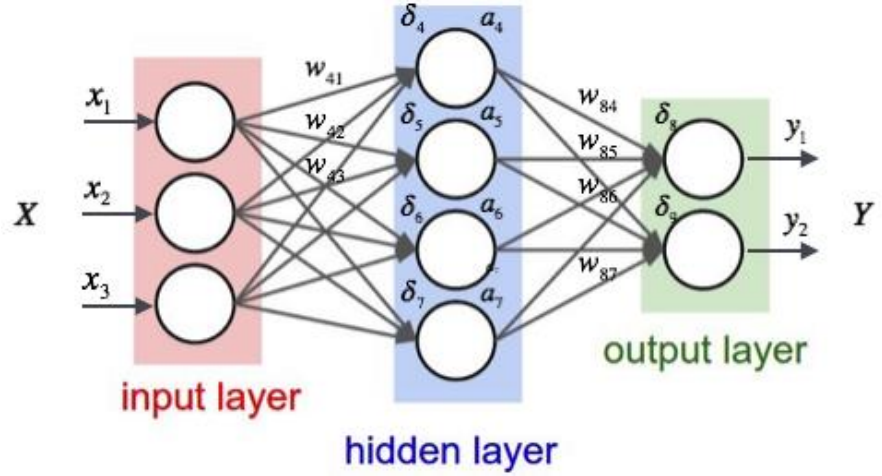


Nevrale nettverk

Klassiske nevrale nettverk er inspirert av hjerneforskning.

Ofte kaller man input og output verdier for “nevroner” eller aktiveringer.





Hvorfor fungerer dyp læring?

For *begynnere* er det ofte bedre å svare på:

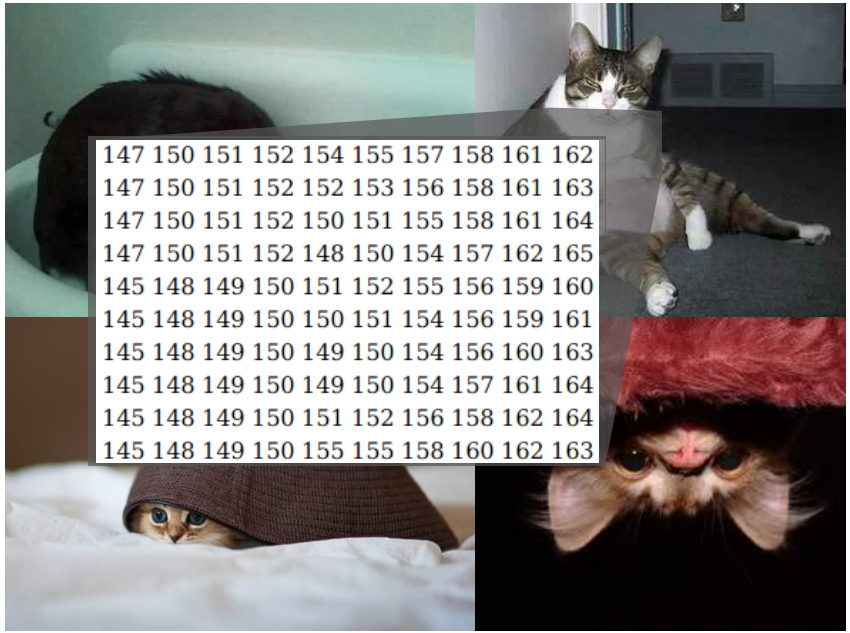
- Hvorfor fungerer ikke dyp læring?
- Hvorfor er maskinsyn/klassifisering vanskelig

For de med *kjennskap* maskinlæring:

- Hvordan kan dette fungere?
- Hvordan kan man lære flere parametere enn man har treningseksempler?

Hvorfor er bildeklassifisering vanskelig?

- Mennesker er gode på bildeklassifisering
- Maskiner ser masse tall
- Stor variasjon



Hvorfor er bildeklassifisering vanskelig?

- Mindre trent på medisinske bilder
- Ikke nødvendigvis vanskeligere

Hvordan kan dyp læring fungere?

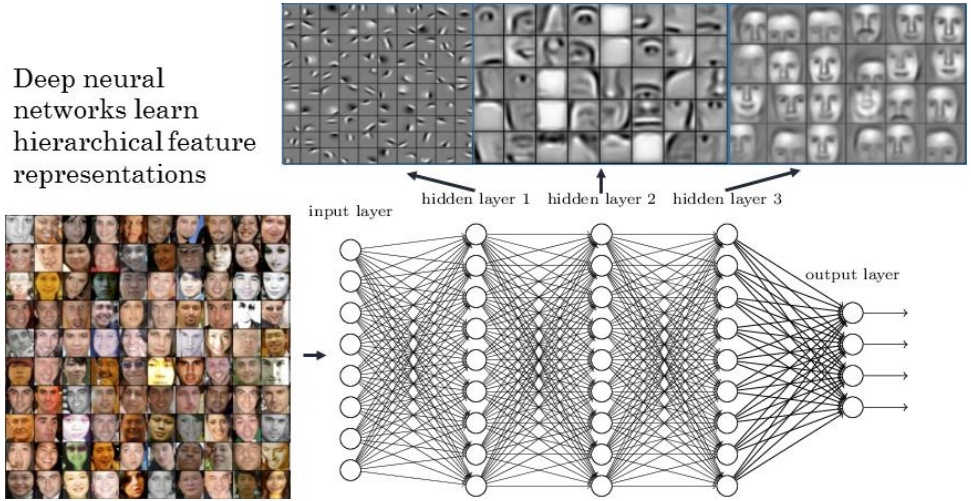
- Ingen kompresjon med egenskapsuttrekning
- Mange flere parametere enn treningseksempler

- Mye handler om å redusere behovet for treningseksempler

Model	image size	# parameters	Mult-Adds	Top 1 Acc. (%)	Top 5 Acc. (%)
Inception V2 [29]	224×224	11.2 M	1.94 B	74.8	92.2
NASNet-A (5 @ 1538)	**299×299**	**10.9 M**	**2.35 B**	**78.6**	**94.2**
Inception V3 [59]	299×299	23.8 M	5.72 B	78.0	93.9
Xception [9]	299×299	22.8 M	8.38 B	79.0	94.5
Inception ResNet V2 [57]	299×299	55.8 M	13.2 B	80.4	95.3
NASNet-A (7 @ 1920)	**299×299**	**22.6 M**	**4.93 B**	**80.8**	**95.3**
ResNeXt-101 (64 x 4d) [67]	320×320	83.6 M	31.5 B	80.9	95.6
PolyNet [68]	331×331	92 M	34.7 B	81.3	95.8
DPN-131 [8]	320×320	79.5 M	32.0 B	81.5	95.8
SENet [25]	**320×320**	**145.8 M**	**42.3 B**	**82.7**	**96.2**
NASNet-A (6 @ 4032)	**331×331**	**88.9 M**	**23.8 B**	**82.7**	**96.2**

Hvordan kan dyp læring fungere?

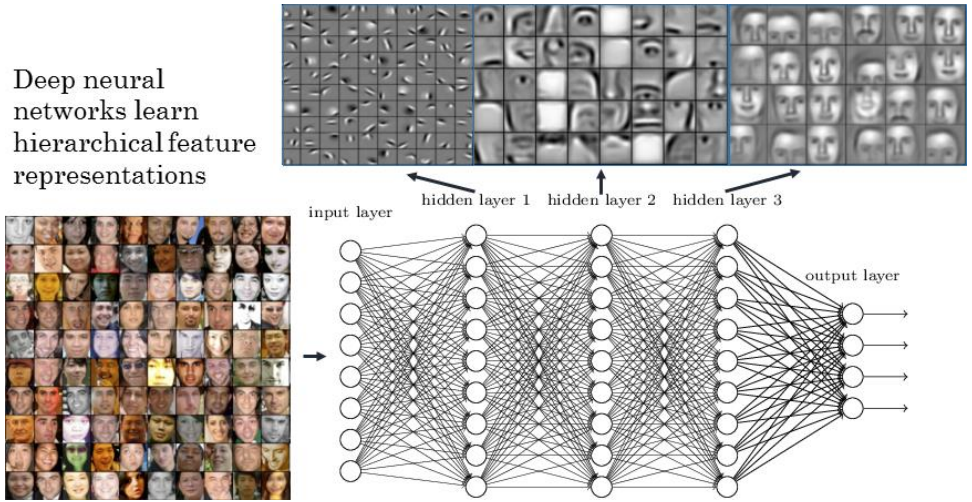
- Tankene til noen av grunnleggerne forklarer navnet
- Dybden og hierarkiet er viktig
- Fortsatt et forskningsområde
    - Selve optimeringsprosessen kan også være viktig
- MYE data hjelper



Hvordan kan dyp læring fungere?

- Deler representasjoner
  - Blobber
  - Kanter
  - Osv.

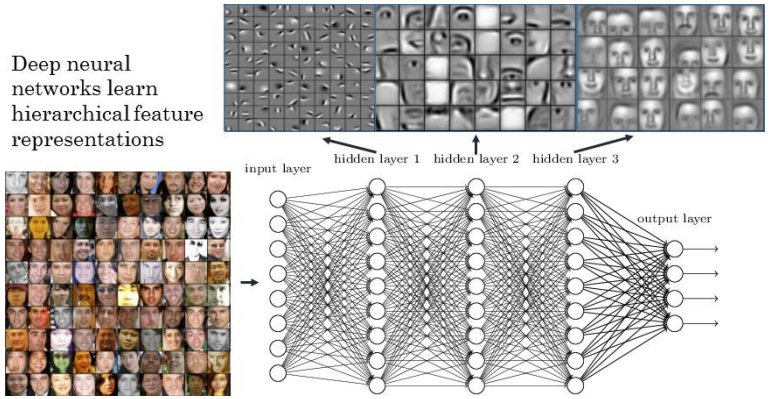
- Gir flere treningseksempler
- Trenger ikke se alle kombinasjoner

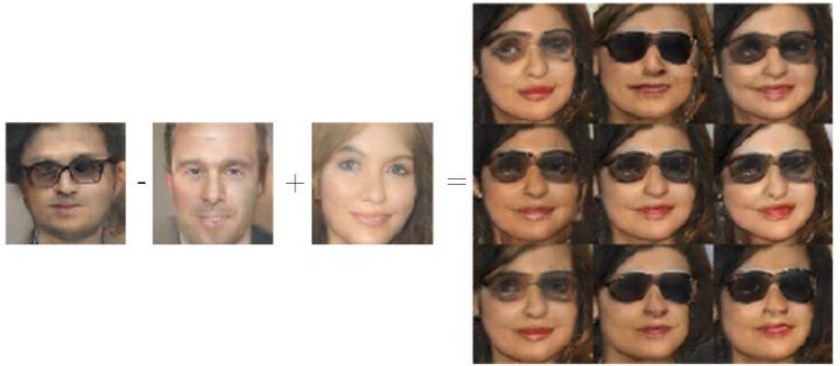


Hvordan kan dyp læring fungere?

- Nevroner som kun reagerer på øye
- Felles for mennesker, løver og kuer

- Studier viser at nett lærer isolerte konsepter





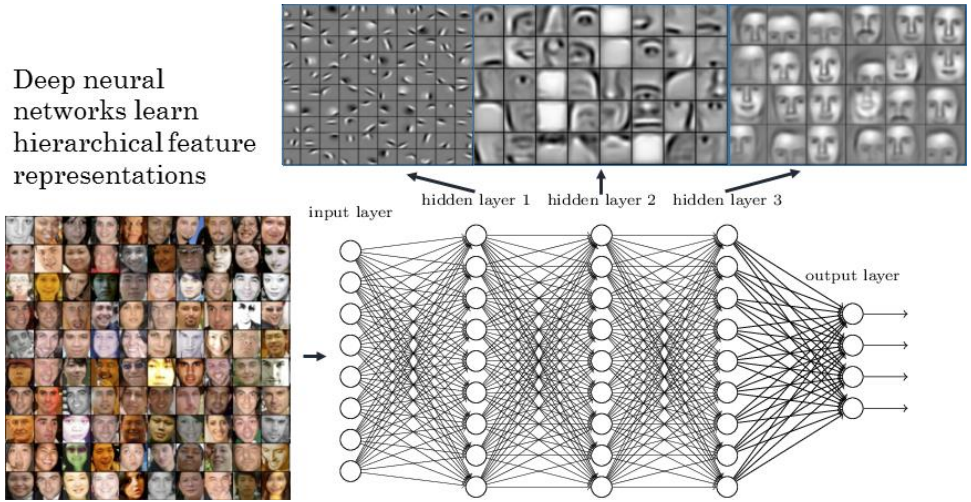
Hva kan et dypt nett lære?

- Kan teoretisk tilnærme alle funksjoner
    - Ikke så praktisk siden man krever uendelig antall vekter
- Et viktigere spørsmål er: Hva er det bra å bruke dype nett til
    - Vi kan ta hint fra teorien om hierarkiske modeller
    - Bildeklasser er typisk hierarkiske
    - Viktig at de lar seg optimere/trene

Hvordan trene et dypt nett?

- Overfladisk gjennomgang
  - Hvordan oppdatere vektene
  - Hvordan ser dette ut
  - Hvordan fungerer dette i et program

- Ingen «instrukser» om interne representasjoner
- Kun et sett kriterier
- Endrer vektene for å



Hvordan tilpasse vektene i et nevralt nett? (Gradient decent)

$L = (f(\mathbf{x}) - y)^2$ f.eks.

1. Definere kriterie (loss funksjon)
   a. f.eks. avstanden fra ønsket verdi til det du fikk ut

$$f = Wx$$

2. Finne gradienten
   a. Hvordan kan man endre vektene minst mulig og få størst forbedring av kriteriet

$$\frac{\partial L}{\partial f} = 2(f(x) - y)\frac{\partial f}{\partial W}$$

1. Oppdatere vektene ved hjelp av gradienten
   a. Endre vekten slik at man får litt bedre loss

$$W \leftarrow W - \alpha \nabla W$$

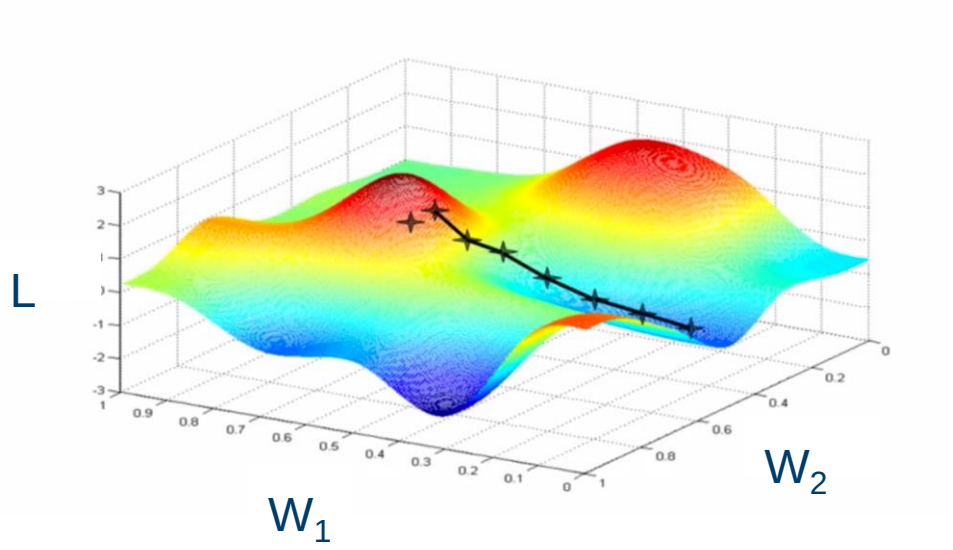
Gradient decent - visuelt

Enkelt nett:

- f(x) = W**x**
- W = $(x, y)^T$

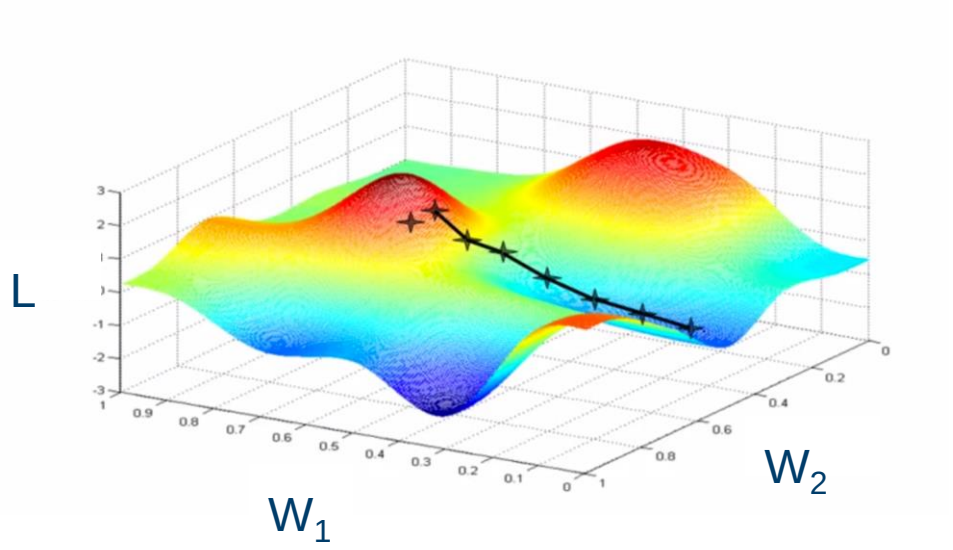
Vi plotter «losset» ved forskellige verdier for W

Følger den bratteste retningen nedover

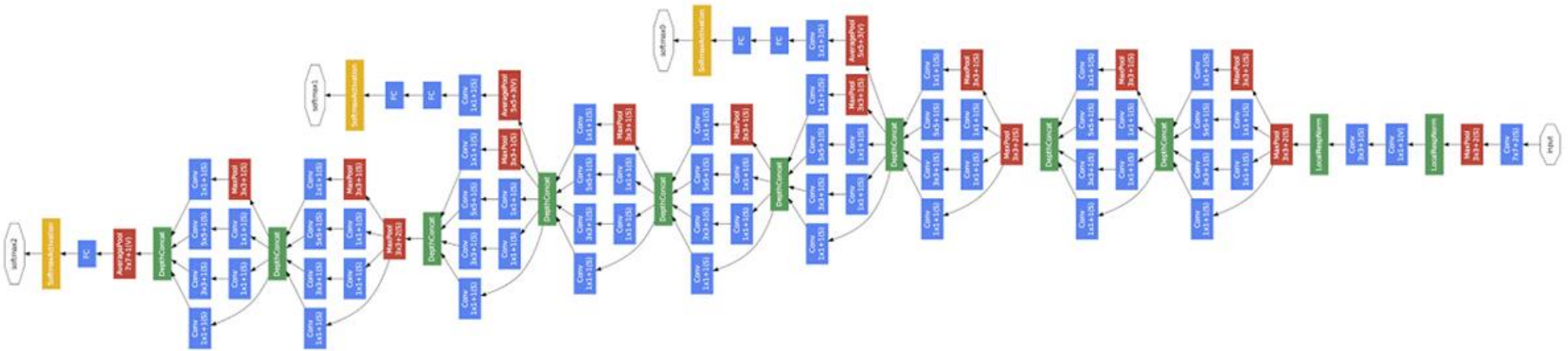


Stocastic **Gradient decent - visuelt**

- Reelt loss?
- Mulighet: gjennomsnitt av alle verdier
- I praksis: ett sett med eksempler for hvert steg



Hvordan finne gradienten til kompliserte uttrykk? Backpropagation



Hvordan finne gradienten til kompliserte uttrykk? Backpropagation

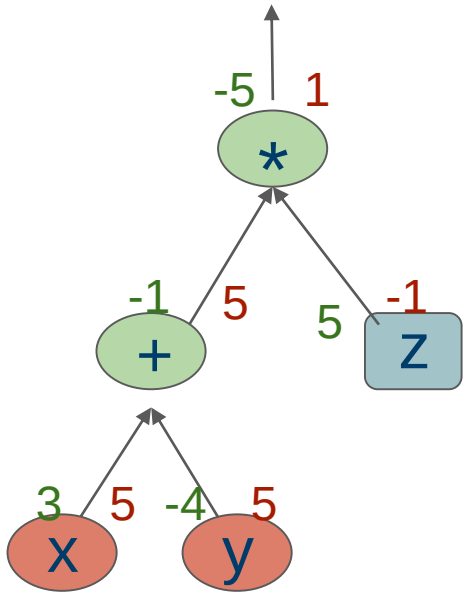
$$f(x,y,z) = (x+y)z$$

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Kjerneregel:

$$F'(x) = f'(g(x))g'(x)$$



Vil vi ikke overtilpasse ett eksempel?

- For å unngå overtilpassning finner vi losset for mange eksempler samtidig
- Bruker gjennomsnittet av gradientene
- Alle eksemplene tar for mye minne

Konvolusjonsnett

Forholdet mellom piksler er ofte viktigere enn absolutt posision

- Bryr vi oss om translasjon
- Det er fortsatt en katt

Katt har flyttet seg - løsning?

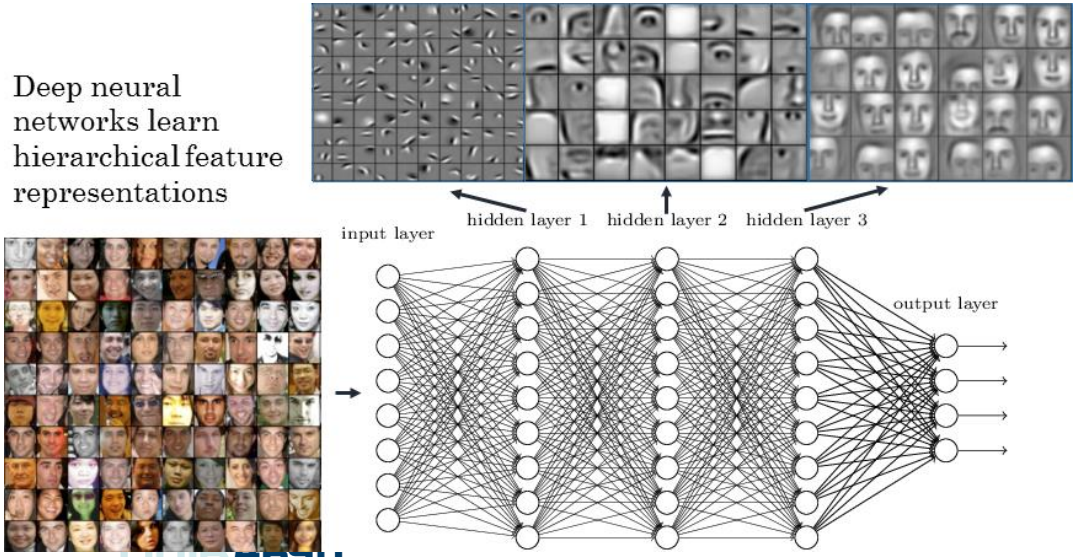
Flytte klassifikatoren rundt i bildet.

Problemer:

- Hva hvis bare øyet har flyttet seg?
- Størrelse på synsfelt
- Lite effektivt

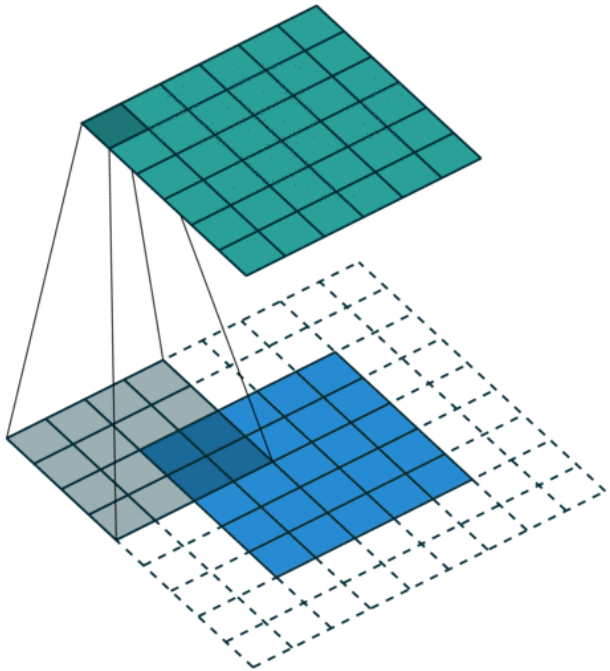
Katt har flyttet seg - la alle lag flytte seg uavhengig

- La hvert lag med sine “detektorer” flytte på seg
- Gjenbruker representasjonene både:
  - i andre lag høyere opp
  - med posisjon
- Blir en multipliserende effekt



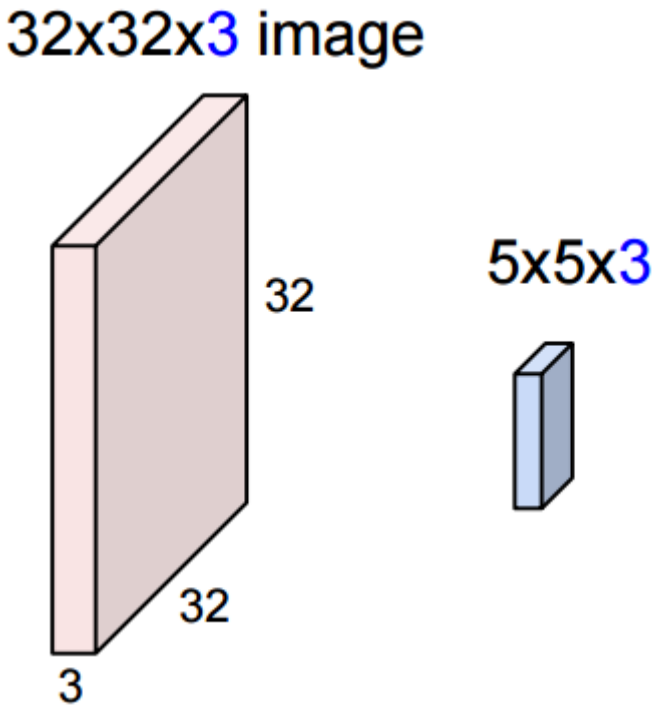
Konvolusjoner

- Bytte matrisemultiplikasjon med konvolusjon
- Legger filter/kjerne over bildet og multipliserer med “vekten” på samme posisjon og summerer alt



Konvolusjoner

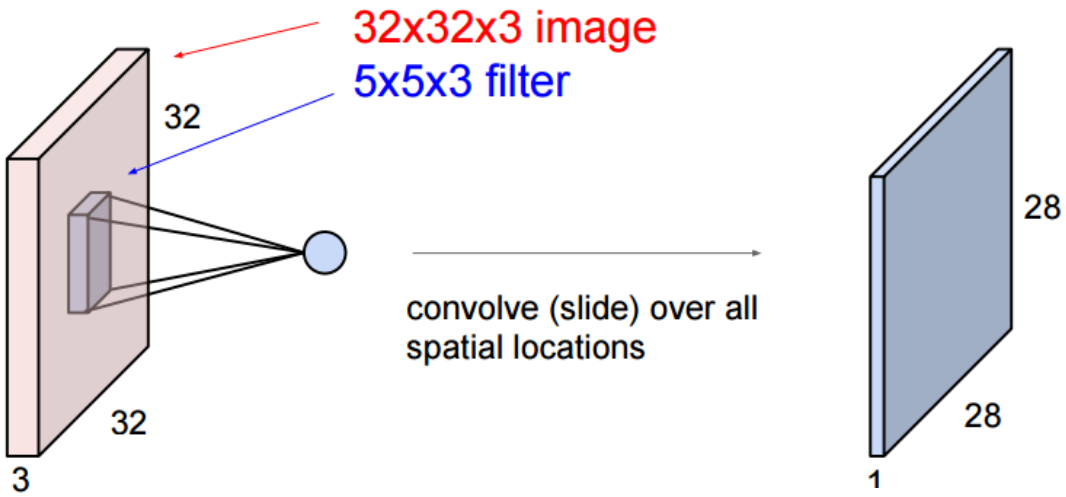
- Bilde og filter har ofte 3 dimensjoner
  Høyde, bredde og kanaler (RGB)
- Fungerer akkurat på samme måte

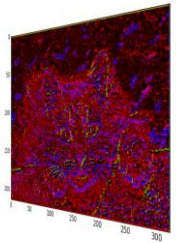


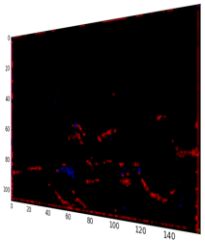
Konvolusjoner

- Overlappende området blir multiplisert, så summert (som et dot produkt)
- Dette gjør man for hver posisjon av filteret

- Ut får man et nytt bildet, men med bare en kanal

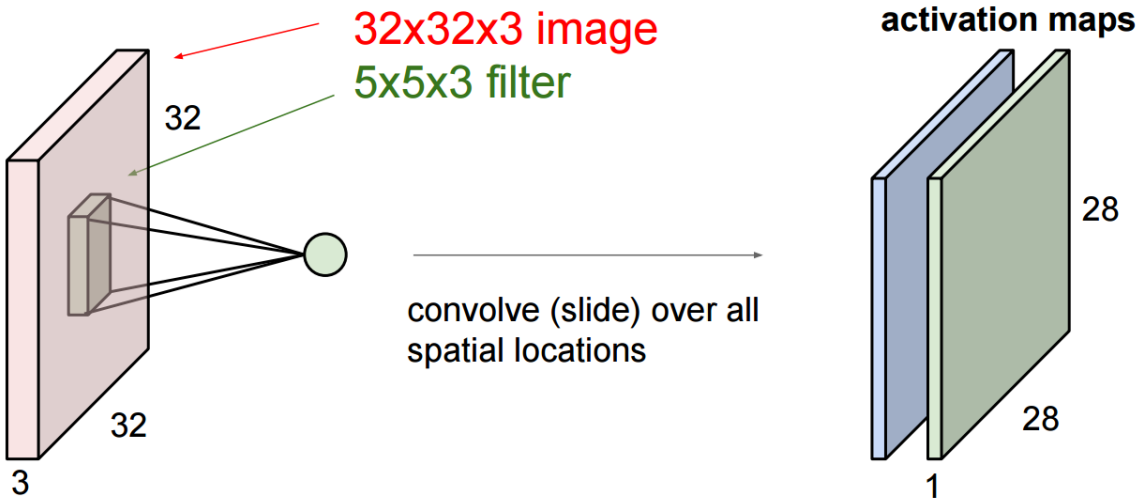






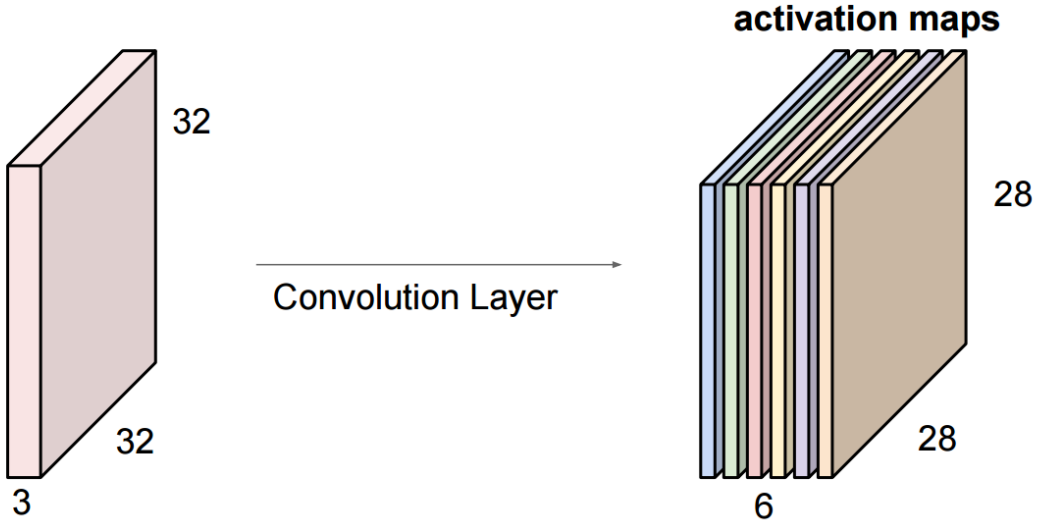
Konvolusjoner - mange filter per lag

- Vi glir en ny kjerne over det samme bildet
- Vi får et nytt bilde ut



Konvolusjoner - mange filter per lag

- Bruker vi 6 filtere får vi et nytt bilde med 6 kanaler, istedet for RGB



Ett lag, to filter nettverk

Input Volume (+pad 1) (7x7x3)

x[:,:,0]

0	0	0	0	0	0	0
0	0	0	1	0	2	0
0	2	1	1	2	2	0
0	0	1	0	0	1	0
0	2	0	2	2	0	0
0	2	0	0	1	0	0
0	0	0	0	0	0	0

x[:,:,1]

0	0	0	0	0	0	0
0	0	2	0	1	1	0
0	1	0	1	1	1	0
0	0	2	1	2	0	0
0	1	0	0	0	1	0
0	0	0	1	0	1	0
0	0	0	0	0	0	0

x[:,:,2]

0	0	0	0	0	0	0
0	0	0	2	0	1	0
0	1	1	1	0	1	0
0	0	0	1	0	0	0
0	0	0	0	2	0	0
0	2	2	1	1	0	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

w0[:,:,0]

-1	0	-1
0	0	0
0	1	1

w0[:,:,1]

1	-1	0
0	1	-1
0	-1	1

w0[:,:,2]

-1	-1	1
-1	1	1
0	0	1

Bias b0 (1x1x1)

b0[:,:,0]

1

Filter W1 (3x3x3)

w1[:,:,0]

-1	1	0
0	1	1
-1	-1	0

w1[:,:,1]

1	1	-1
0	0	-1
0	0	-1

w1[:,:,2]

1	1	-1
-1	-1	-1
0	1	1

Bias b1 (1x1x1)

b1[:,:,0]

0

Output Volume (3x3x2)

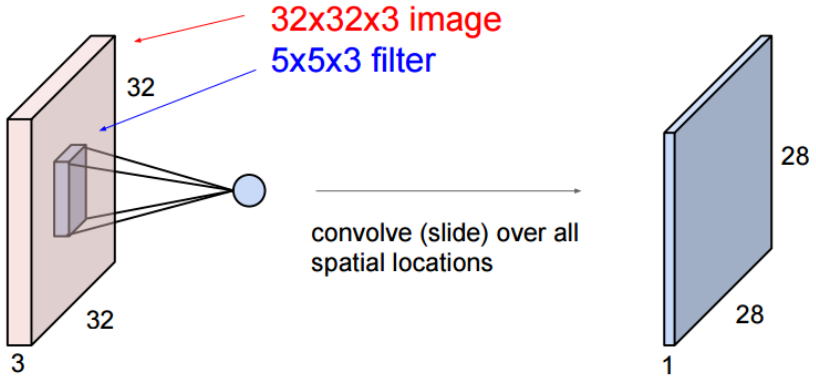
o[:,:,0]

2	5	4
-2	1	-3
4	2	-4

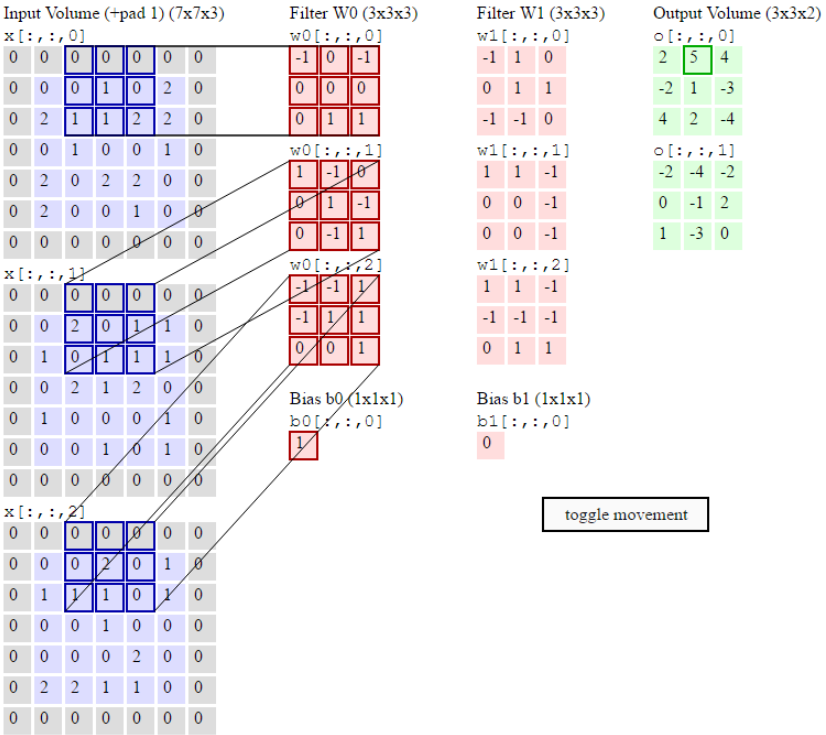
o[:,:,1]

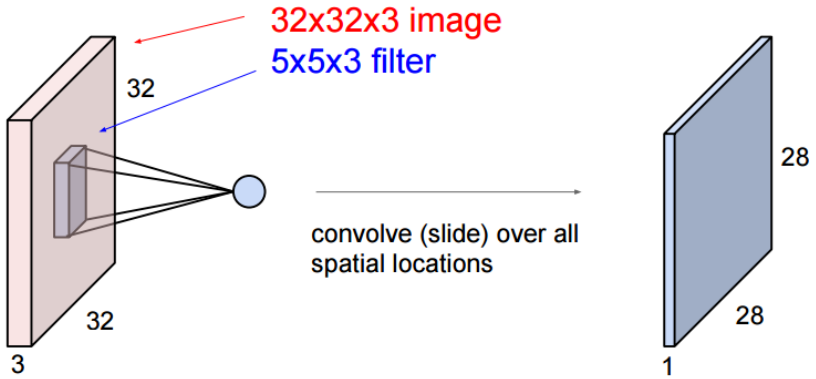
-2	-4	-2
0	-1	2
1	-3	0

toggle movement

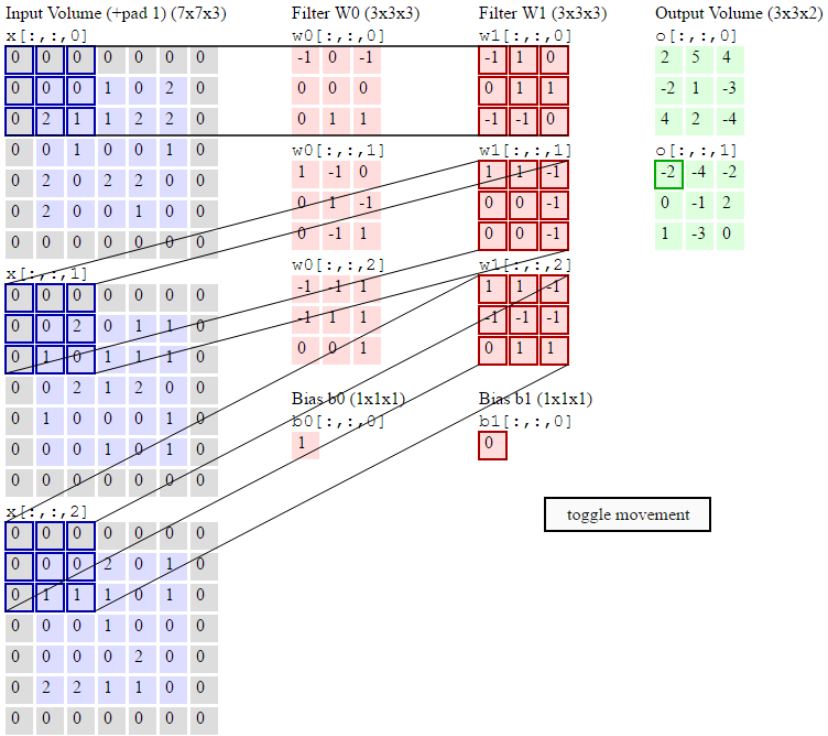


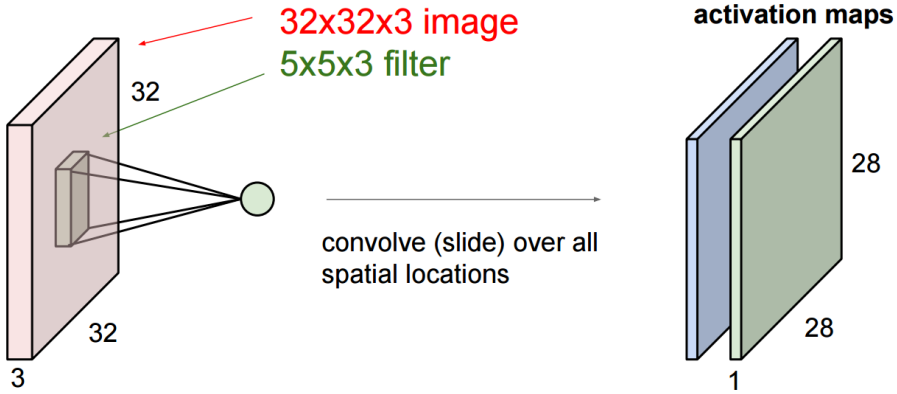
Ett lag, to filter nettverk



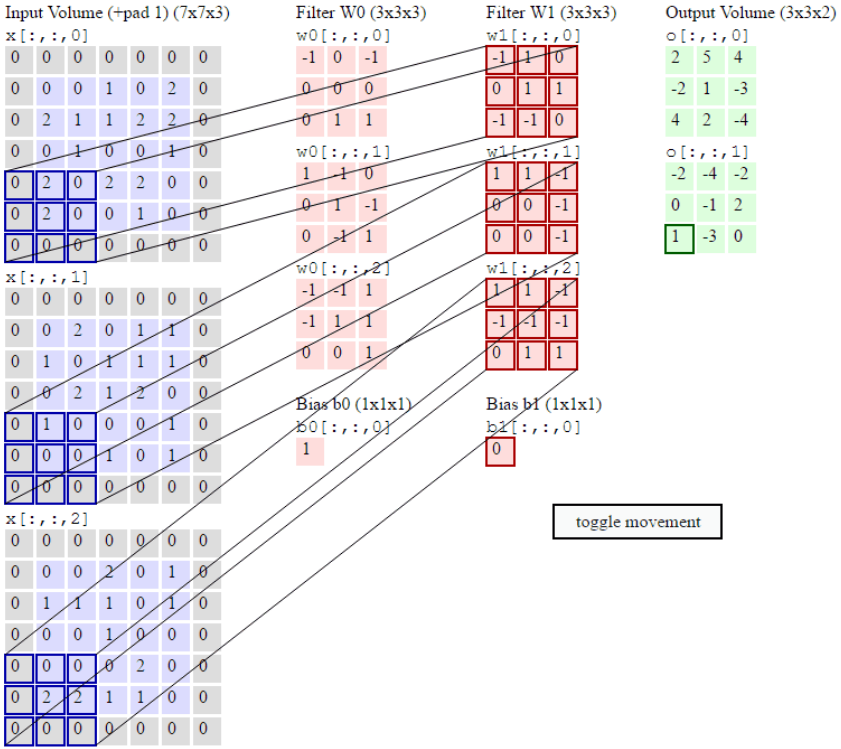


Ett lag, to filter nettverk



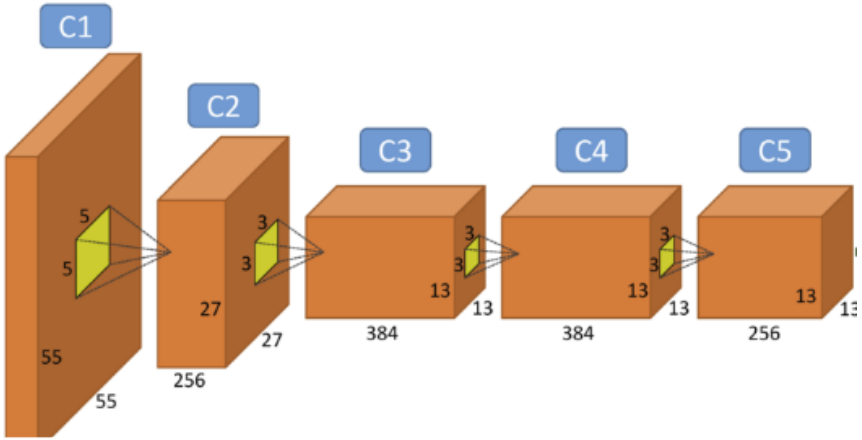


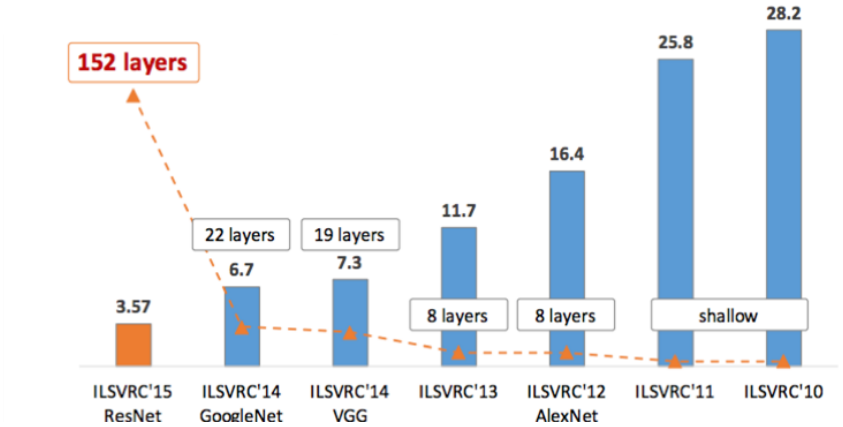
Ett lag, to filter nettverk



Flere lag...

Kan se omtrent slik ut.





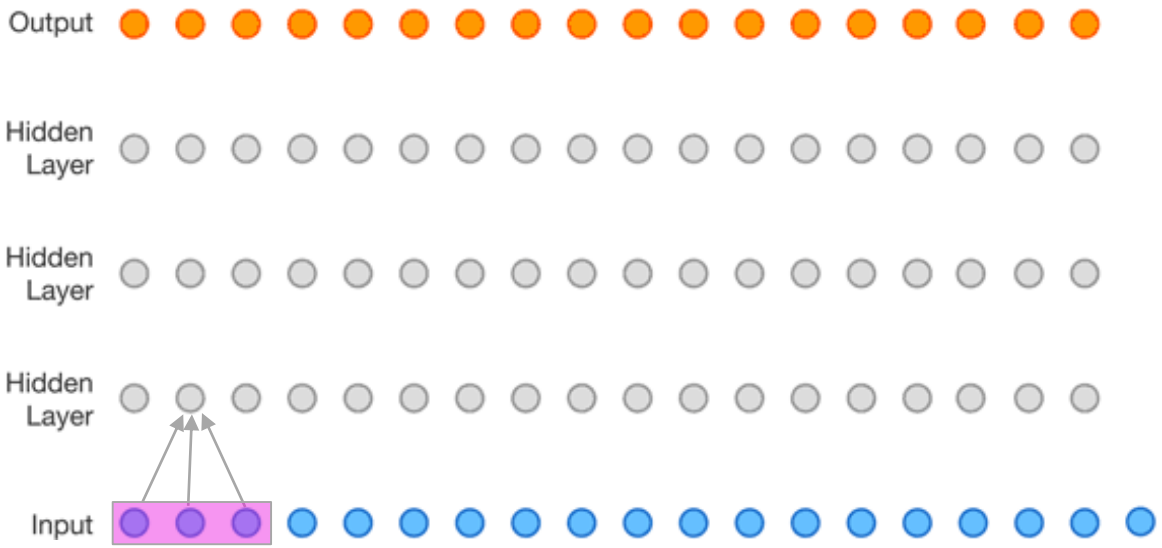
Konvolusjonsnett - synsfelt

Hvor stort området påvirker resultatet

- Med en glidende klassifikator, så får du inputen som **synsfelt**
- Trenger vi større?

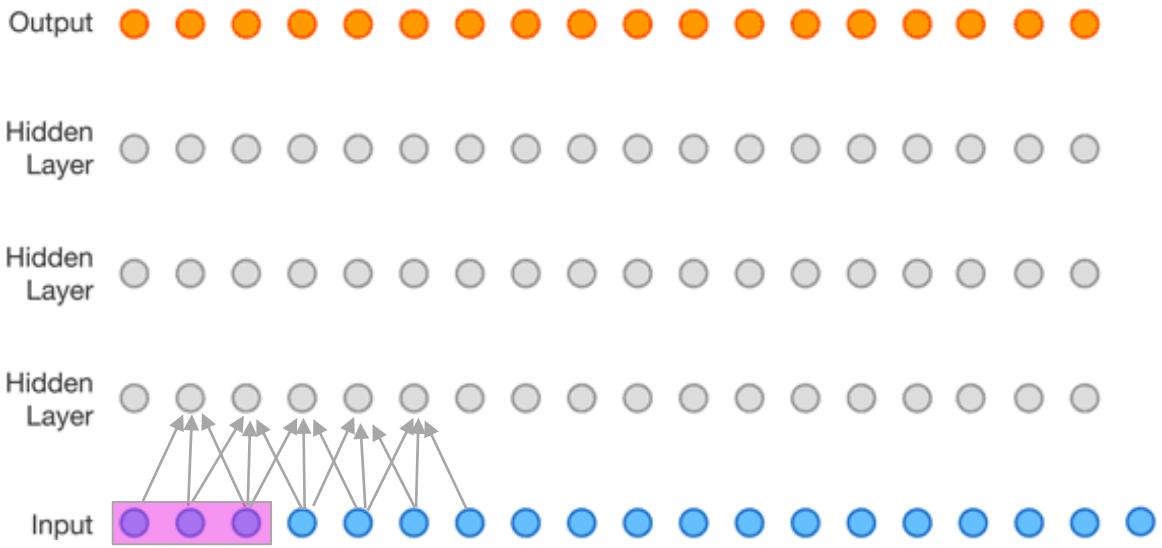
Hvor stort området påvirker resultatet?

- For konvolusjonsnettverk så vokser **synsfeltet** for hvert lag



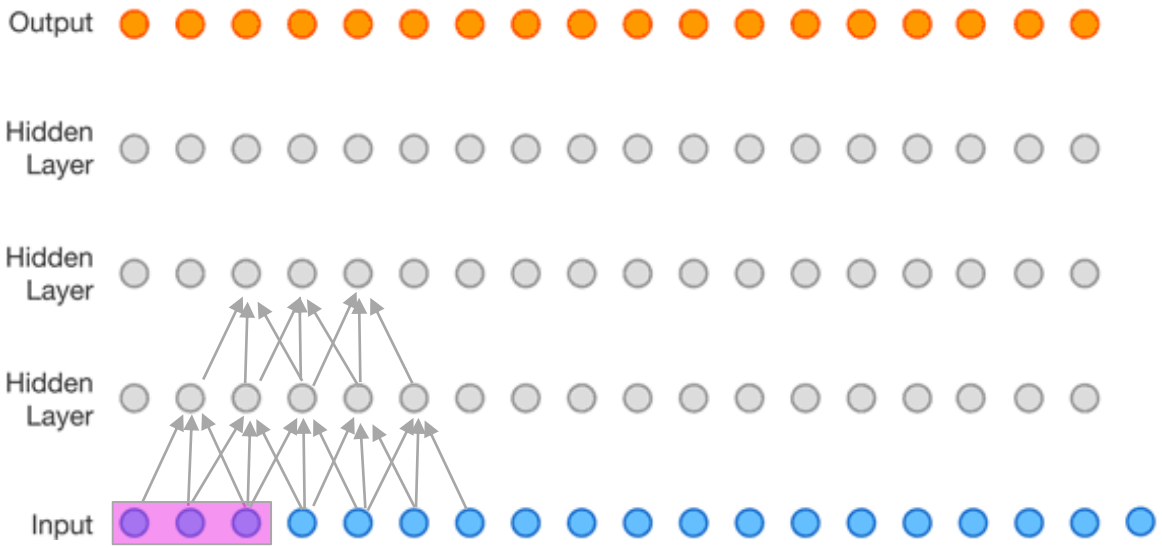
Hvor stort området påvirker resultatet?

- For konvolusjonsnettverk så vokser **synsfeltet** for hvert lag



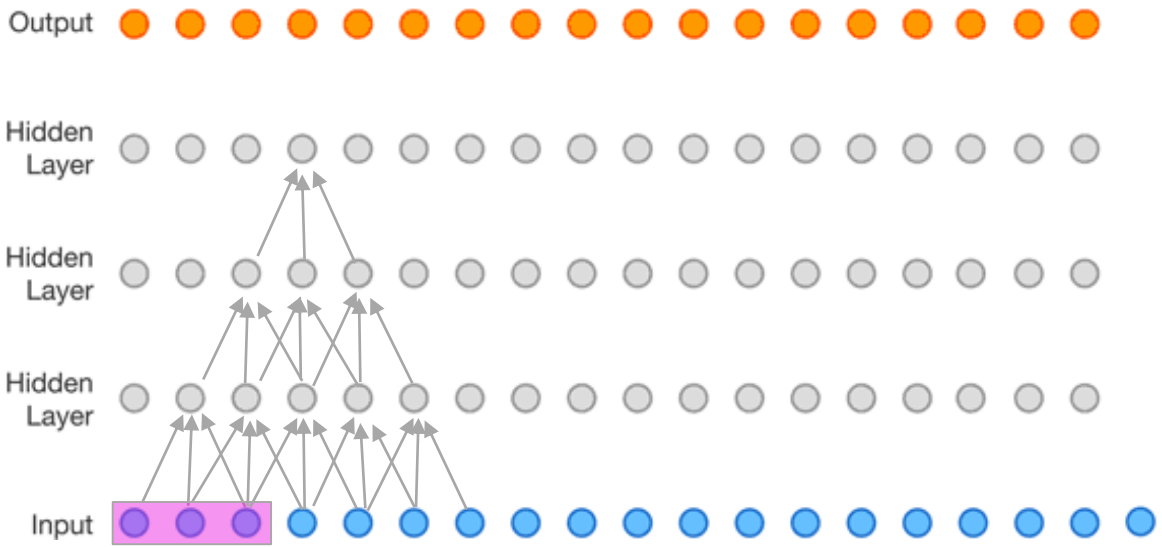
Hvor stort området påvirker resultatet?

- For konvolusjonsnettverk så vokser **synsfeltet** for hvert lag.

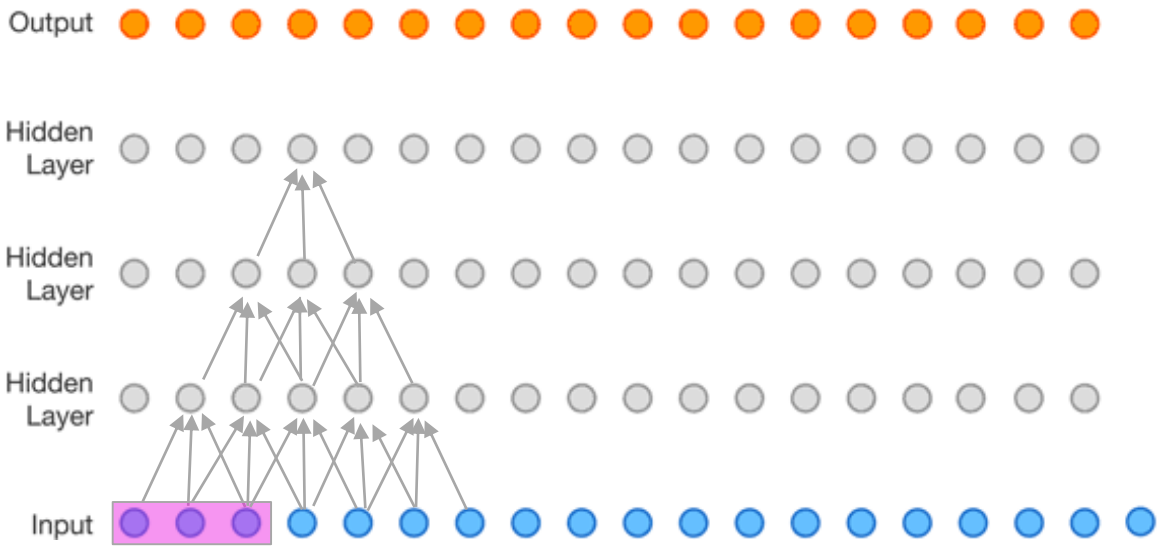


Hvor stort området påvirker resultatet?

- For konvolusjonsnettverk så vokser **synsfeltet** for hvert lag

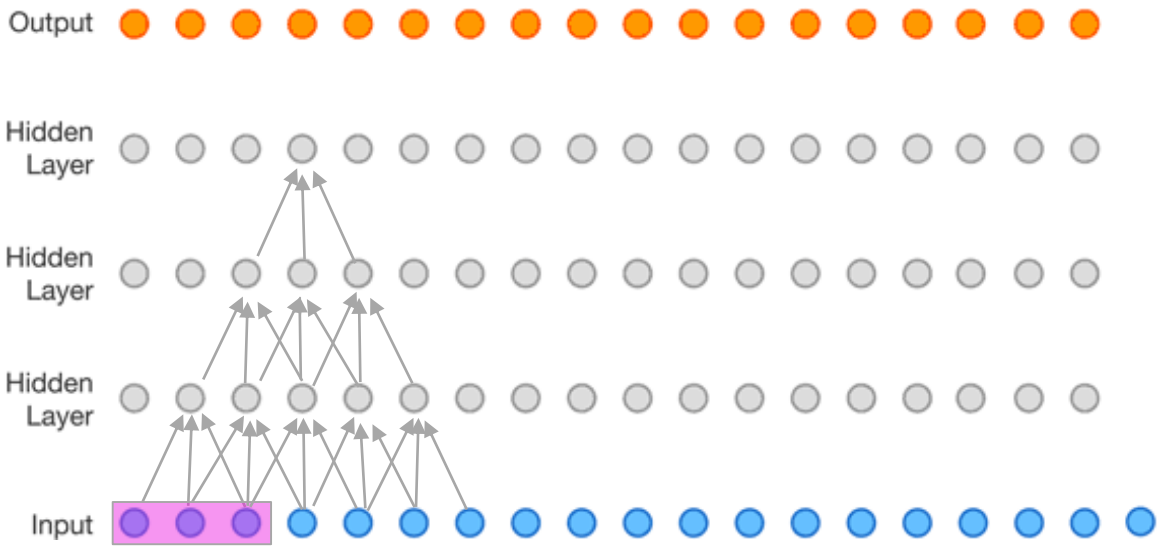


Hvor stort området påvirker resultatet?



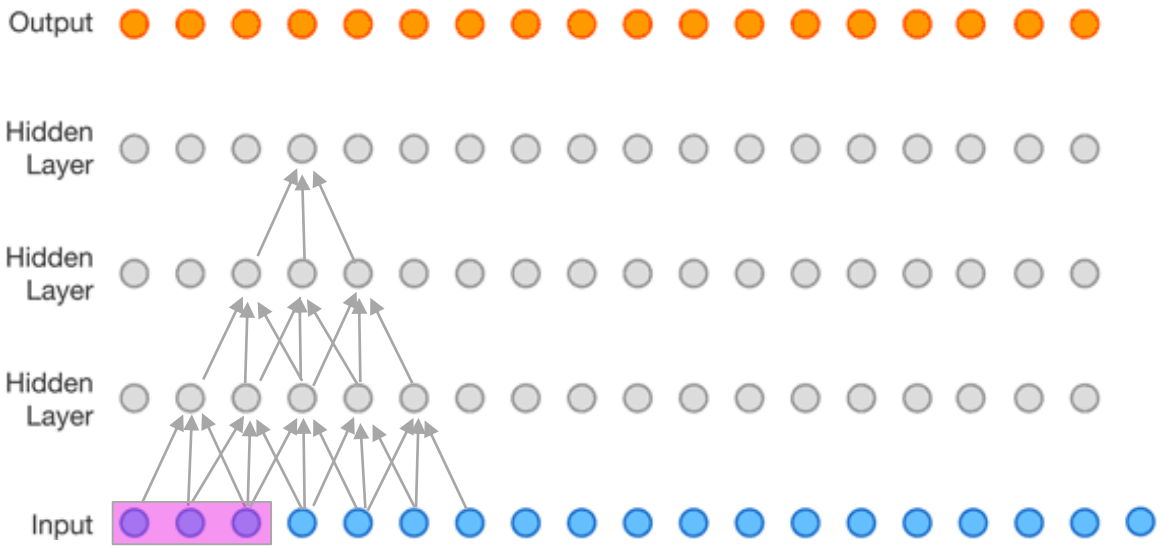
Synsfeltet vokser med k-1 for hvert lag

Hvor *k* er filterstørrelsen



Stort filter eller mange lag?

Små filtre er mer effektive med tanke på antall vekter.



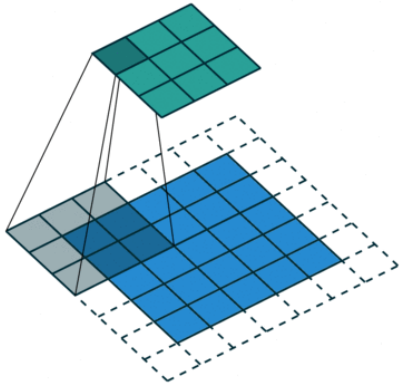
Hvordan øke synsfeltet mer effektivt

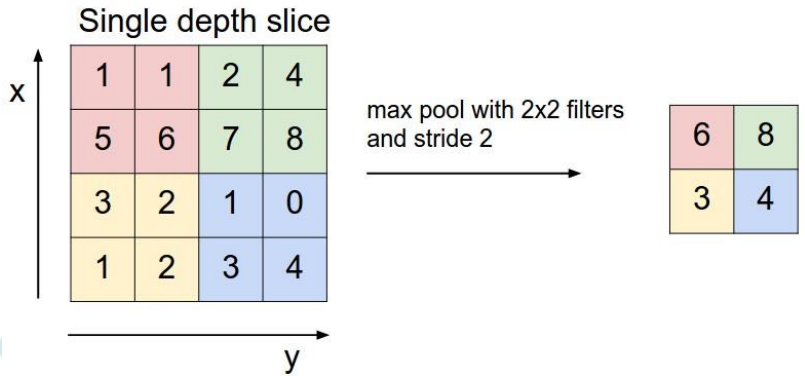
“strided konvolutions” (hoppende konvolusjoner)

- Mest vanlig
- Enklest

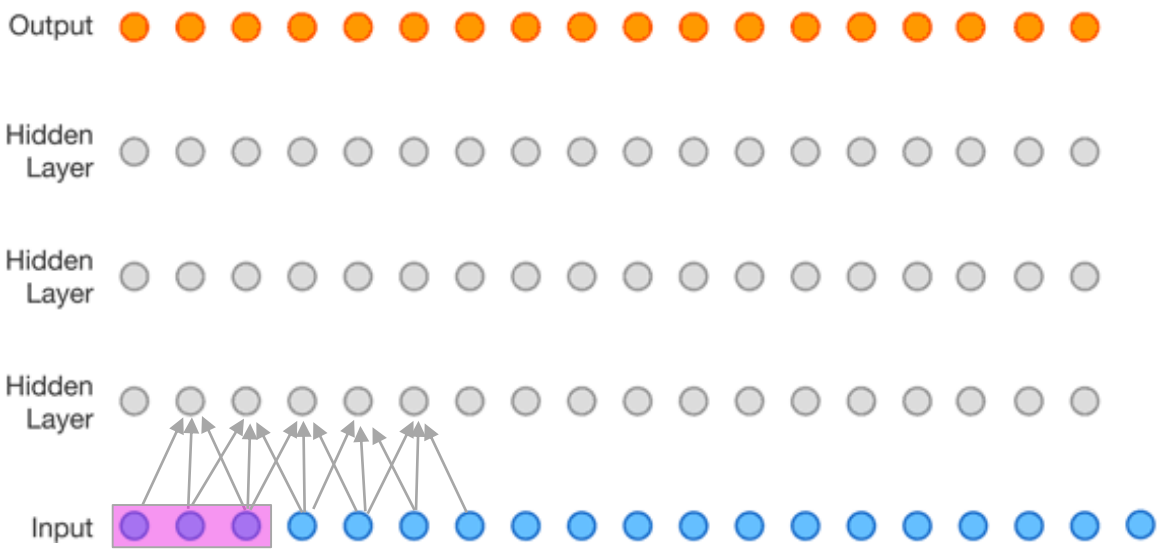
Alternativer er forskjellige former for “pooling”

- max-pool
- average-pool



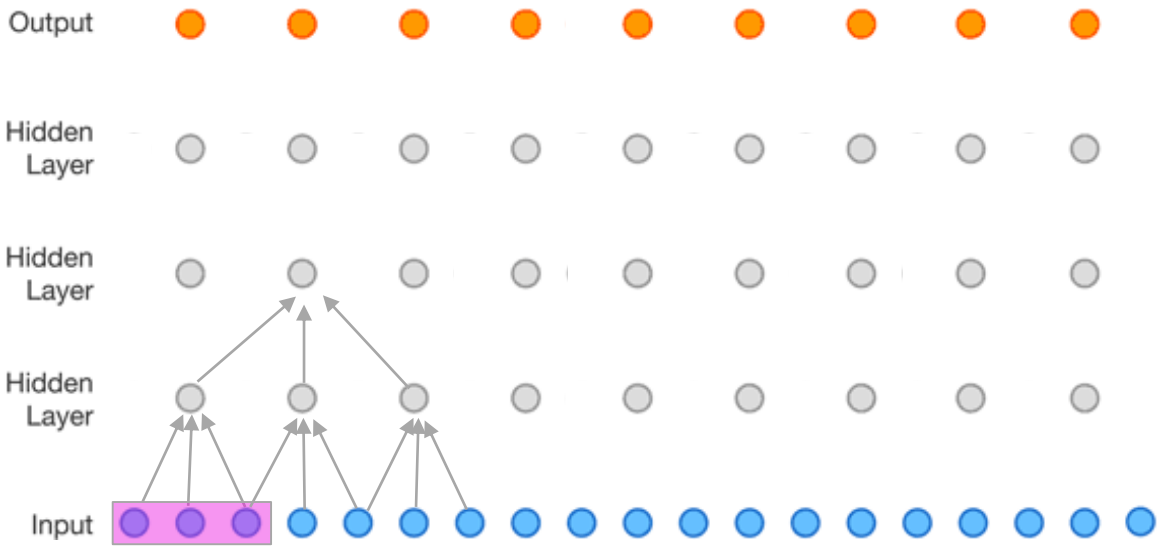


Effekten av hopp



Effekten av hopp

- Vi dekker hele inputen (ikke hull)
- Vi ser et større området

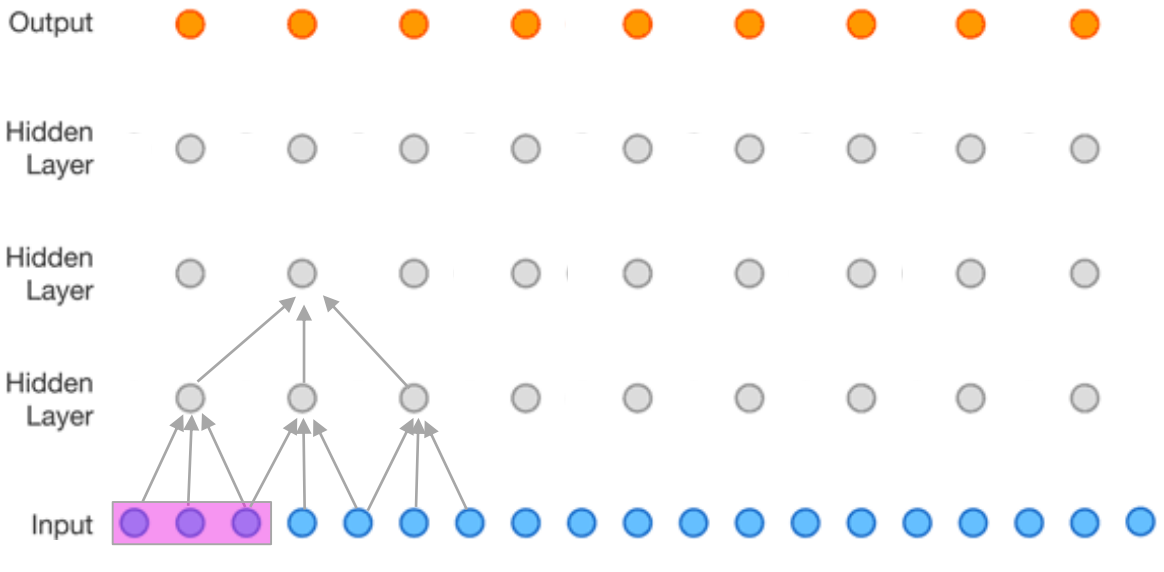


Effekten av hopp

Vi kan se effekten ved hjelp av en funksjon l_k.

l_{k-1} er forrige synsfelt f er filterstørrelsen og s_i er størrelsen på hoppet i steg i.

$$l_k = l_{k-1} + ((f_k - 1) * \prod_{i=1}^{k-1} s_i)$$



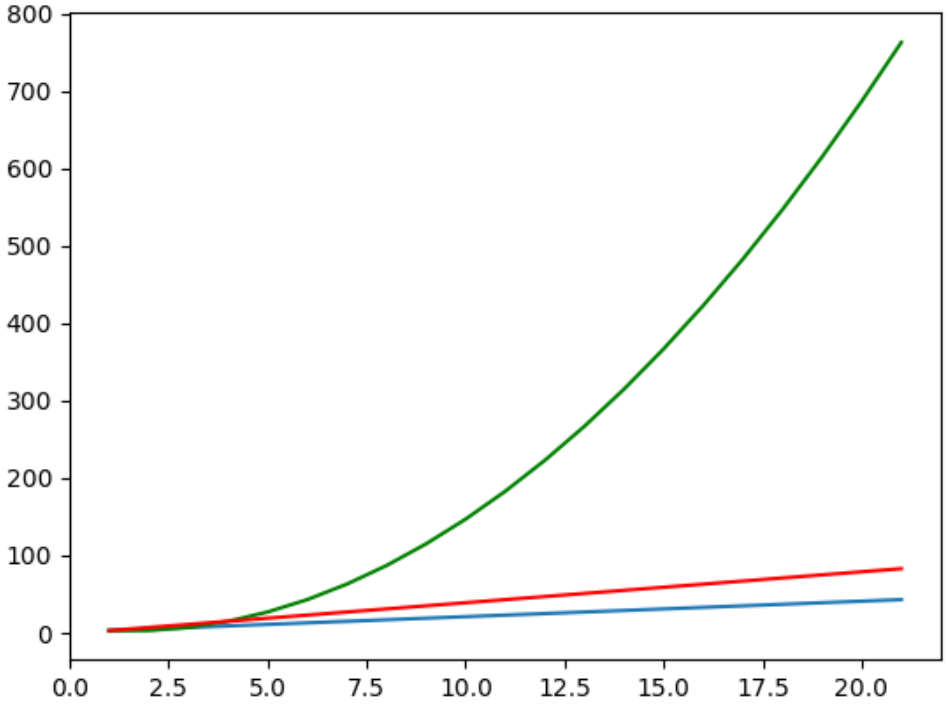
Effekten av hopp

$$l_k = l_{k-1} + ((f_k - 1) * \prod_{i=1}^{k-1} s_i)$$

Alle de på følgende lagende vil få synsfeltet multiplisert med **S.**

Lag nummer er på x-aksen og **synsfelt** på y-aksen.

Grøn: hopp= 1, **Rød:** hopp=1, kun i det andre laget, **Blå:** hopp=0



Hvorfor tenke på synsfelt

- Hva kan man forvente at et nettverk ser?
- Hvor stort nettverk trenger man?

Lurt å tenke på

Hvorfor trener ikke nettverket? Null-gradient

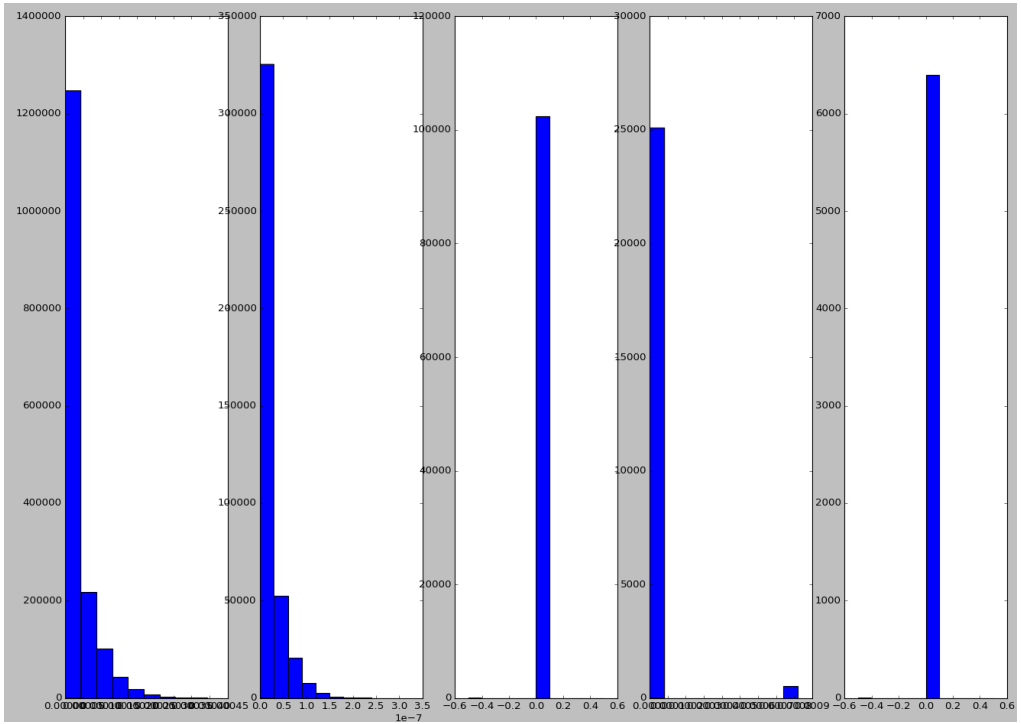
1. Hvis input er null: Vektene i det laget blir ikke trent
2. Gradienten fra senere lag er null: Vektene i det laget og alle senere lag blir ikke trent.
3. Vektene er null: Vektene i alle tidligere lag blir ikke trent

$$dW = X^T O$$

$$dX = WO^T$$

Lignende problem med norm av vekter < 1

For hvert lag blir verdiene mindre og mindre, og man får problemer med null verdier.



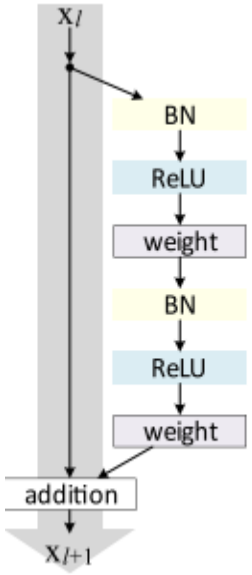
Mulige løsninger

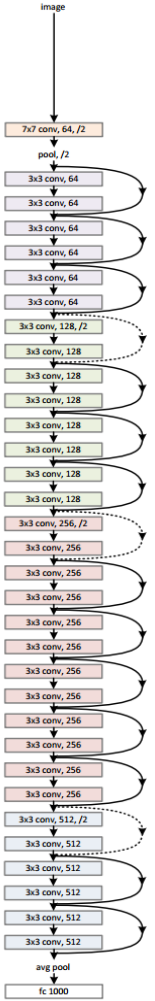
- Batch normalisering
  - Normalisering for hvert lag
- ResNet / skip-koblinger
- Nøye initialisering

Typiske bruksområder

Typisk nettverk

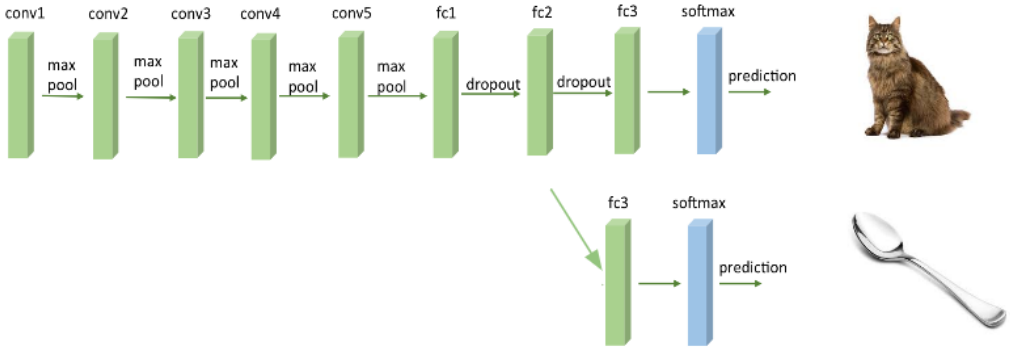
- ResNet har blitt standard arkitektur
- Batch norm og skip kobling gjør det enklere å trene





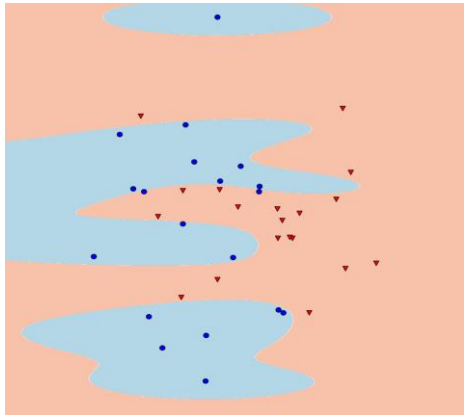
Ferdigtrente nettverk

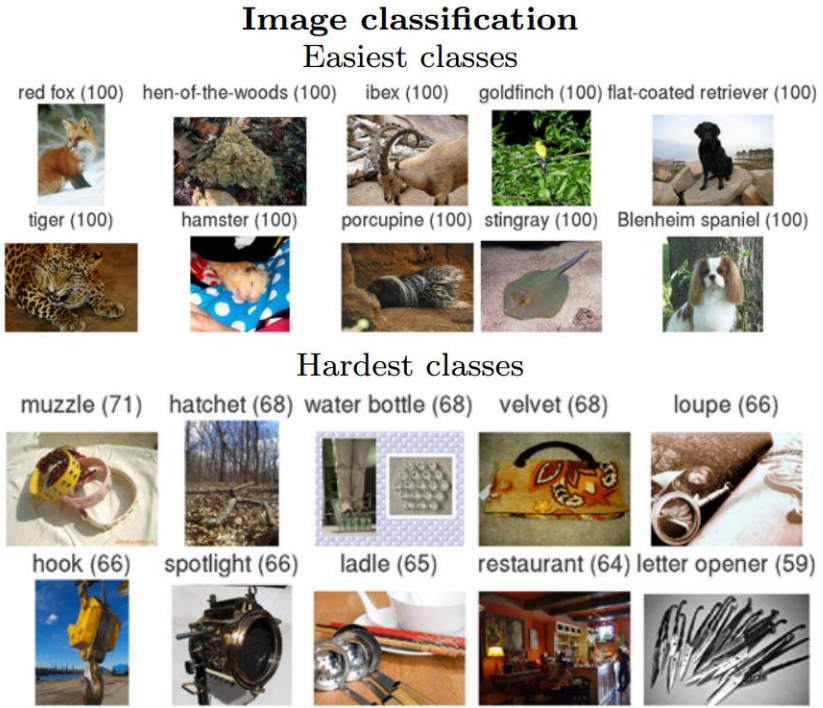
- Vekter kan gjenbrukes for forskjellige formål
- Kan kreve mindre treningsdata



Klassifisering

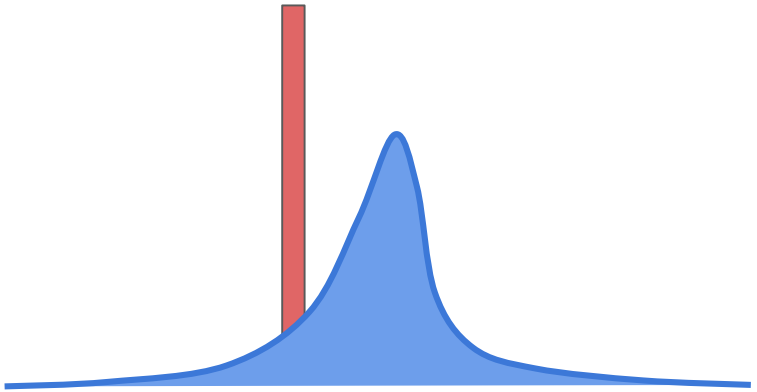
- ImageNet
- Mange oppgaver kan formuleres som klassifiserings oppgaver
  - Sjekke om noe er tilstede
  - Telle noe
  - Ta beslutninger





Klassifisering

- Typisk softmax loss
- Gir ut “log likelihood” for hver klasse
- Gi høyest mulig sannsynlighet for riktig klasse



$$\sigma(x_j) = \frac{e^{x_j}}{\sum_i e^{x_i}}$$

$$L(x) = \sum_i -y_i log(\sigma(x_i))$$

Segmentering

Knytte enkelt piksler til objekter

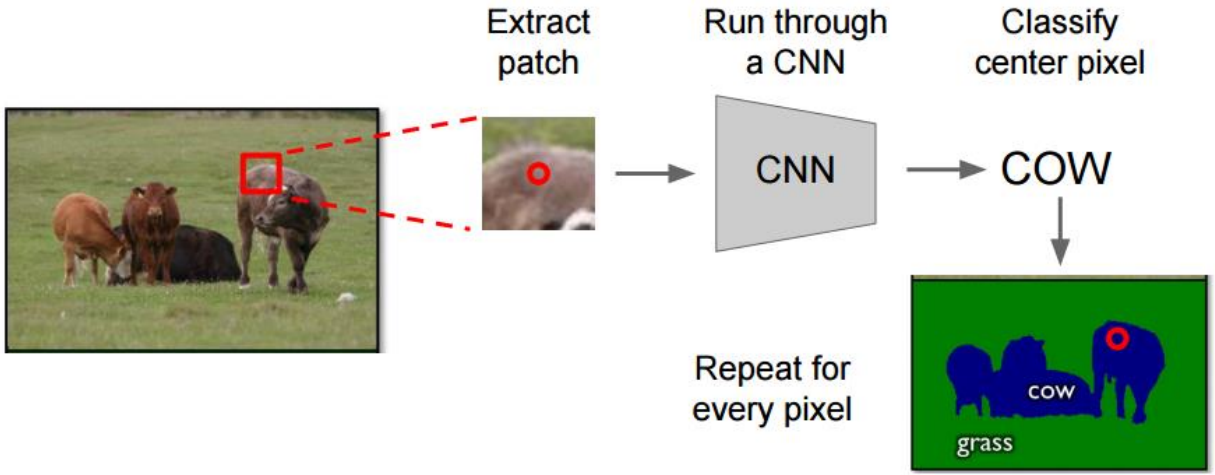
MSCOCO er standard benchmark

Kan brukes til:

- Finne ut hvor man kan kjøre
- Måle kalorier i mat
- Måle hjertevolum
- osv.

Segmentering

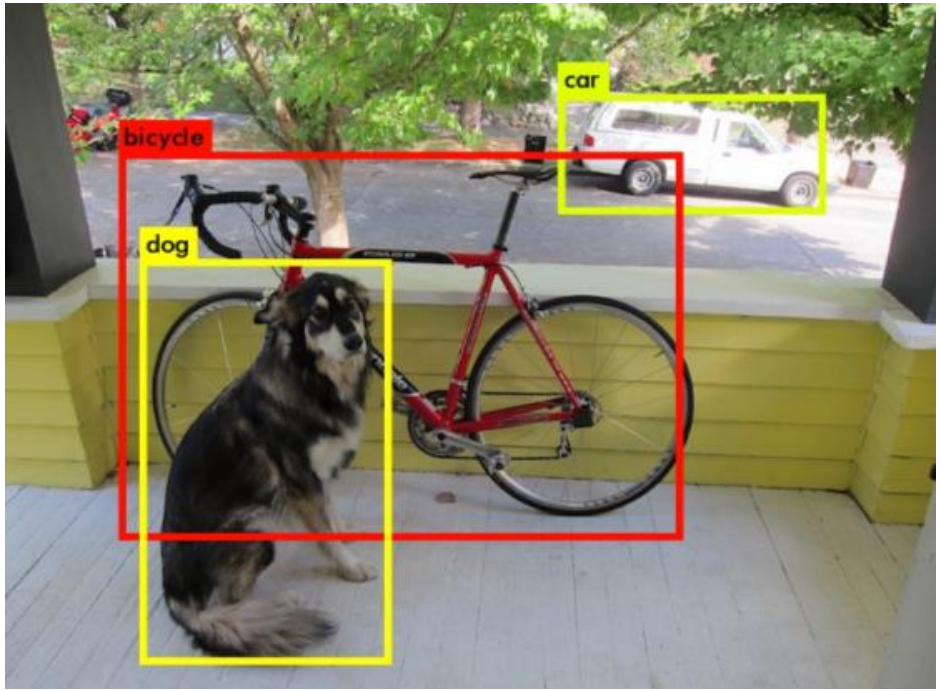
Klassifiserer simpelt nok hver piksel.



Deteksjon

- Finne objekter (separat)
- Når man ikke har segmentering
- Brukt som en del av enkelt eksempel segmentering

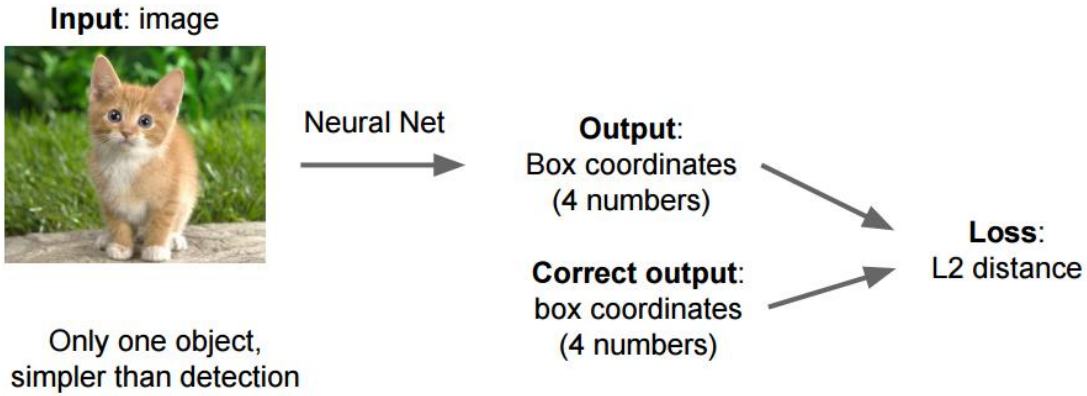
You only look once: Unified, real-time object **detection**



Deteksjon

- Finn bokser med regressjon L2 loss
- $(f(x) - y)^2$ på bokskoordinater
- Kan kombineres med klassifisering

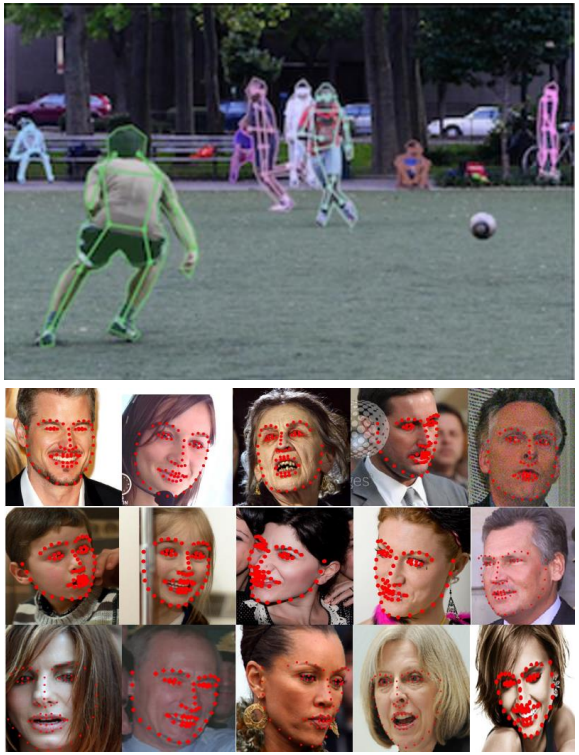
You only look once: Unified, real-time object **detection**



Keypoint detection

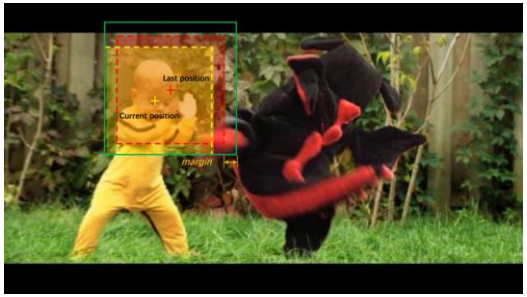
Finne kjente punkter

Towards Accurate Multi-person Pose Estimation in the Wild



Tracking, re-identification and odometry

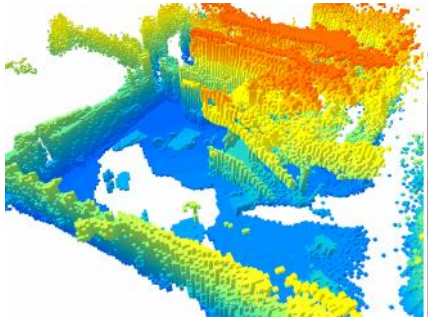
- Følge eller gjenkjenne “vilkårlige” objekter eller punkter
- Gjenkjenne mennesker
- Lage kart fra en video, ved matching av piksler

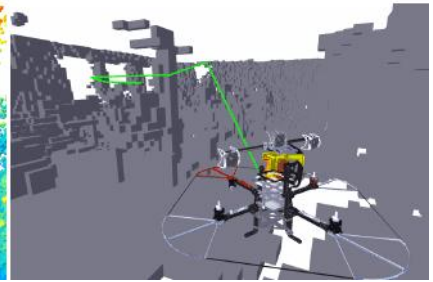


Dual Deep Network for Visual Tracking

DeepVO: A Deep Learning approach for Monocular Visual Odometry

A Multi-task Deep Network for Person Re-identification





Tracking, re-identification and odometry

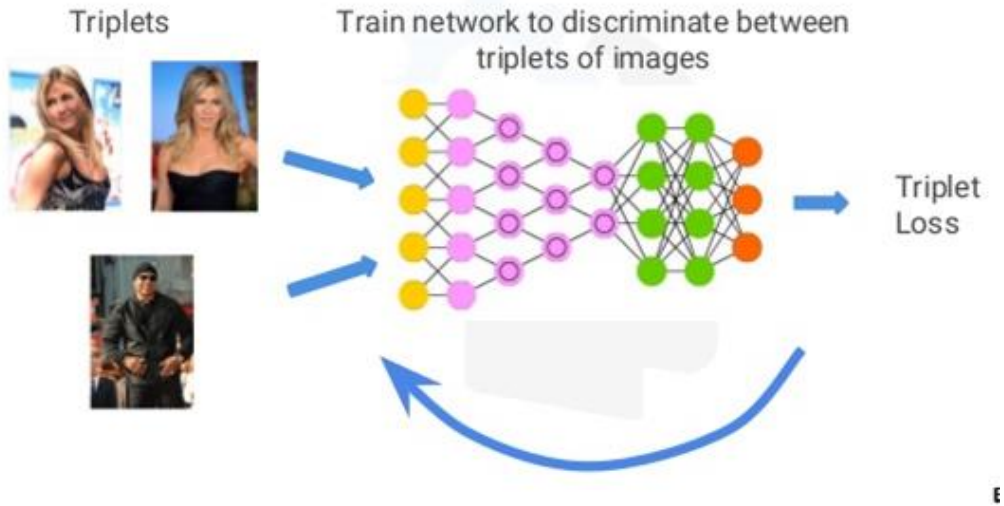
“Triplet loss”:

- Tvinger “like” til å få lik output
- Forskjellige til å få forskjellig output

Dual Deep Network for Visual Tracking

DeepVO: A Deep Learning approach for Monocular Visual Odometry

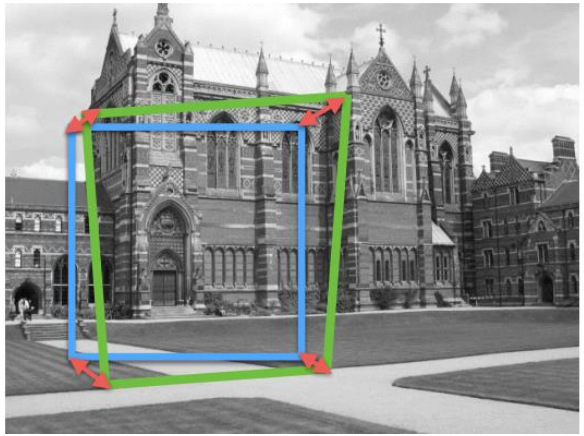
A Multi-task Deep Network for Person Re-identification



Posisjon/Homografi estimering

Direkte estimering av projeksjons matrise

[Deep Image Homography Estimation]



Tilslutt

- Dype nett har mange bruksområder
- Ofte verdt å vurdere om dette er en mulig løsning
- Å merke data er ofte mindre jobb en man tror
- Å forsøke nye modeller/bruksområder kan kreve mye jobb og kunnskap
- Ikke alle problemer er lett å løse på denne måten