



Computer Vision Training

21-02-2020



Millennials.ai

Millennials.ai is a research based Artificial Intelligence (A.I.) company with a focus on middle large companies already using their data to efficiently execute their business proposition. They integrate custom made algorithms and machine learning into existing business processes to optimize the value of your business data.





Wie ben ik?

- MSc Artificial Intelligence aan Universiteit van Amsterdam
- Natural Language Processing Engineer bij Attendi



Inhoudsopgave

1. Introductie
2. Deel I
 - Image Filtering
 - Network Layers
 - Convolutional Neural Networks
 - Data tekort
 - Assignment 1
3. Deel II
 - Introduction to Generative Models
 - Variational Autoencoders
 - Generative Adversarial Networks
 - Assignment 2

Introductie



Wat is Computer Vision?

1. Computer

Elektronische machine die bepaalde processen en operaties kan uitvoeren op basis van een set instructies, gestuurd door hardware of software

2. Vision

Het begrijpen van een lokale omgeving door de belichting van objecten door het lightspectrum

3. Computer Vision

- *Machines die proberen te begrijpen wat ze zien om een bepaald doel te bereiken*
- *Het proces waarbij een machine of systeem een begrip van visuele informatie krijgt door (een) algoritme(s) te runnen op de gegeven informatie*



Voorbeelden van applicaties

- Optical character recognition: Afbeeldingen van tekst vertalen naar een vorm die kan worden begrepen door een machine
Handgeschreven postcodes op brieven herkennen
- Object detection: Objecten identificeren in foto's en video's
Tellen van auto's
- Fingerprint recognition: Pattern information van een vingervinprint gebruiken om een vergelijking te maken tussen een source en target vingervinprint
Automatische toegangsverlening



Computer Vision Approaches

1. Bottom-up

- Gebruikt het begrip van verzamelde informatie om verder begrip te ontwikkelen over een arbitraire observatie
- Al het verzamelde begrip leidt naar een oplossing of een algemeen begrip van het geheel van de observatie

2. Top-down

- Gebruikt achtergrondkennis om begrip te produceren over een observatie
- Achtergrondkennis wordt gebruikt als referential guide om parameters te selecteren die een model passen

Computer Vision Approaches

Automatisch begrijpen van foto's en video's:

- Bottom-up: Eigenschappen berekenen van de 3D wereld van visuele data
Image processing, 3D reconstruction
- Top-down: Algoritmes en representaties die computers objecten, mensen, scènes en activiteiten laten herkennen
Segmentation, image classification, object detection

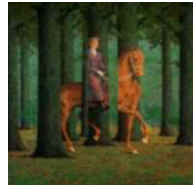


Wat maakt Computer Vision ingewikkeld?

- Viewpoint variation



- Occlusion



- Appearance change



- Illumination change



- Background clutter

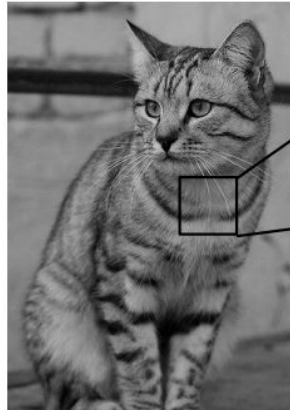


- Orientation and scale



Wat is een zwart-wit afbeelding?

- Afbeelding bestaat uit 3 dimensies
Hoogte, breedte, aantal kleurkanalen
- Zwart-wit: 1 kleurkanaal
- Elk getal representeert een
bepaalde grijswaarde intensiteit
- $[0, 255]$ / $[0, 1]$



[185	112	100	111	104	99	106	99	96	103	112	119	104	87	93	87]
[91	98	102	106	104	79	98	103	99	105	123	136	118	105	94	85]
[76	85	98	105	120	105	87	96	95	99	115	112	106	103	99	85]
[99	81	81	93	120	131	127	100	95	98	102	99	96	93	101	94]
[106	91	61	64	69	91	88	85	101	107	109	98	75	84	96	95]
[114	100	85	55	55	69	64	54	64	87	112	129	98	74	84	91]
[133	137	147	103	65	81	80	65	52	54	74	84	102	93	85	82]
[128	137	144	140	109	95	86	78	62	65	63	63	68	73	86	101]
[125	133	148	137	119	121	117	94	65	79	88	65	54	64	72	98]
[127	125	131	147	133	127	126	131	111	96	89	75	61	64	72	84]
[115	114	109	123	150	148	131	118	113	109	100	92	74	65	72	78]
[89	93	98	97	108	147	131	118	113	114	113	109	106	95	77	88]
[63	77	86	81	77	79	102	123	117	115	117	125	125	136	115	87]
[62	65	82	89	78	71	80	101	124	126	119	101	107	114	131	119]
[63	65	75	88	89	71	62	81	120	138	135	105	81	98	118	118]
[87	65	71	87	106	95	69	45	76	138	126	107	92	94	105	112]
[118	97	82	86	117	123	116	66	41	51	95	89	89	95	102	107]
[164	146	112	80	82	120	124	104	76	48	45	66	88	101	102	109]
[157	170	157	120	93	86	114	132	112	87	69	55	78	82	99	94]
[130	128	134	161	139	100	109	118	121	134	114	87	65	53	69	86]
[128	112	96	117	158	144	120	115	184	107	102	93	87	81	72	79]
[123	107	96	86	83	112	153	149	122	169	104	75	88	107	112	99]
[122	121	102	80	82	86	94	117	145	148	153	162	58	78	92	107]
[122	164	148	103	71	56	78	83	93	103	119	139	102	61	69	84]]

What the computer sees

Wat is een kleurafbeelding?

- 3 kleurkanalen
- Pixels een combinatie van Rood, Groen en Blauw (RGB)
- 100x100 kleurafbeelding = $100 \times 100 \times 3 = 30,000$ getallen
- 30,000 getallen tussen 0-255: Is dit een hond of een kat?

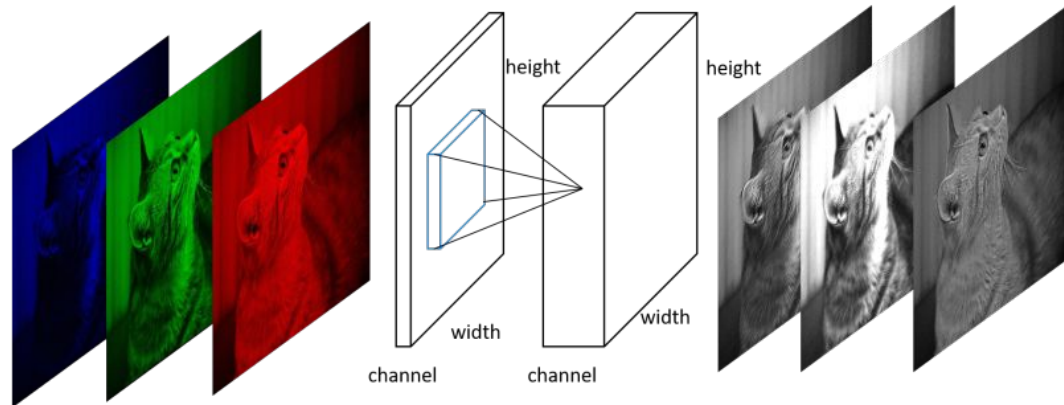
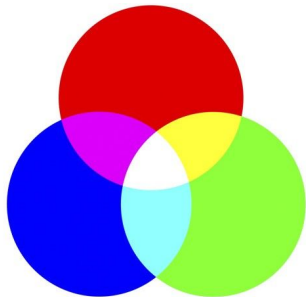
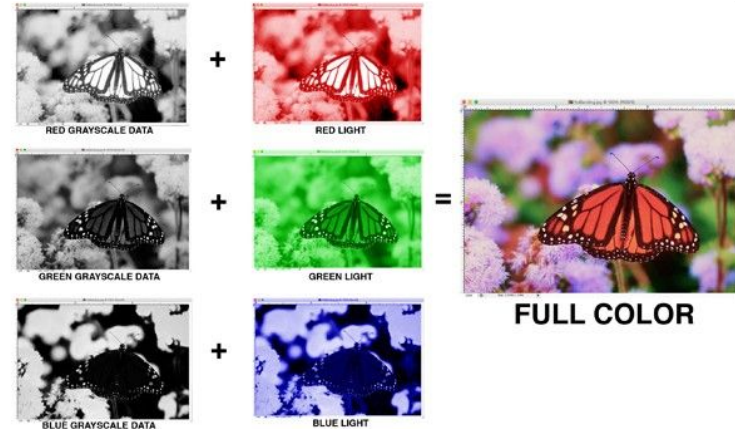
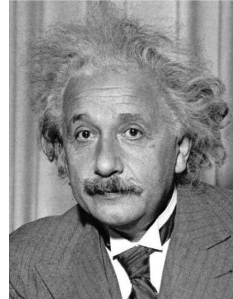
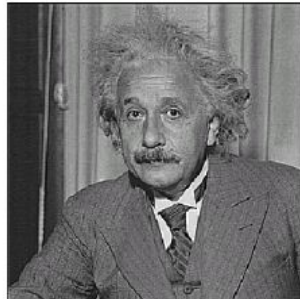
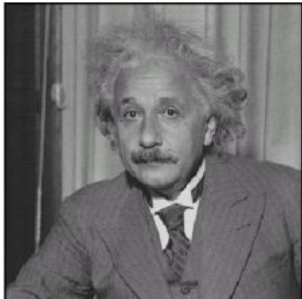


Image Filtering

Applicaties

- Enhancement
Noise reduction, sharpening, resizing
- Information extraction
Textures, edges
- Pattern detection
Template matching





Box Filter

Vervangt iedere pixel met het gemiddelde van zijn neighbourhood

$$\frac{1}{9} g[\cdot, \cdot]$$

1	1	1
1	1	1
1	1	1

$g[\cdot, \cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Box Filter

Reference
point

 $F[x, y]$

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	90	0	90	90	90	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

 $G[x, y]$

Box Filter

$g[\cdot, \cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

0									

Box Filter

$g[\cdot, \cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

0	10								

Box Filter

$g[\cdot, \cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20						

Box Filter

$g[\cdot, \cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30					

Box Filter

$g[\cdot, \cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30				

Box Filter

$g[\cdot, \cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$F[x, y]$

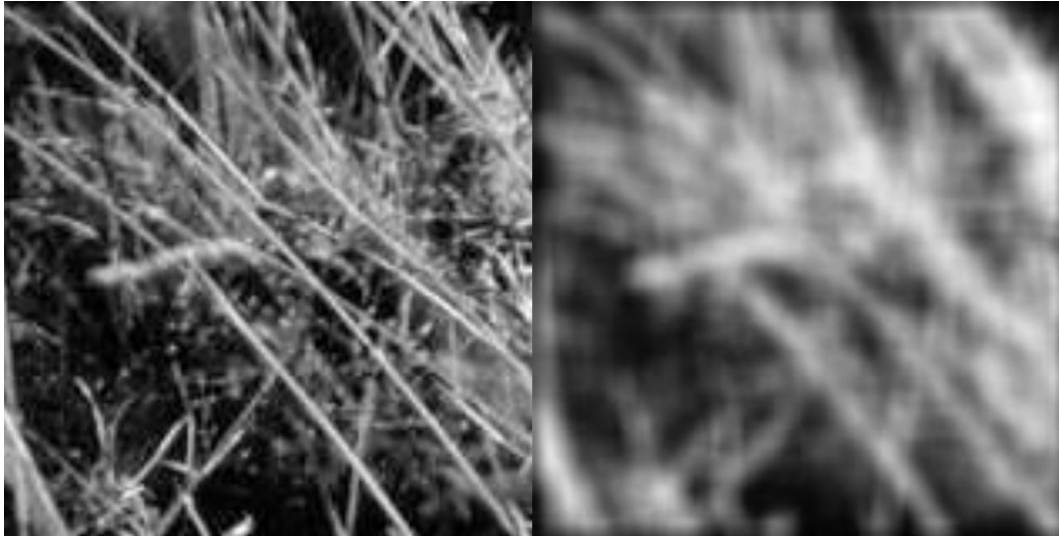
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	



Box Filter



Identity Filter



Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Shifting Filter



0	0	0
0	0	1
0	0	0



Sharpening Filter

Accentueert de verschillen met het lokale gemiddelde



0	0	0
0	2	0
0	0	0

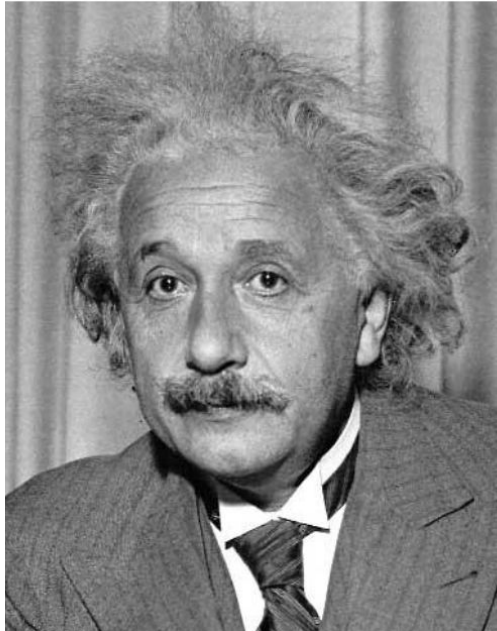
—

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1



Sobel Filter (verticaal)

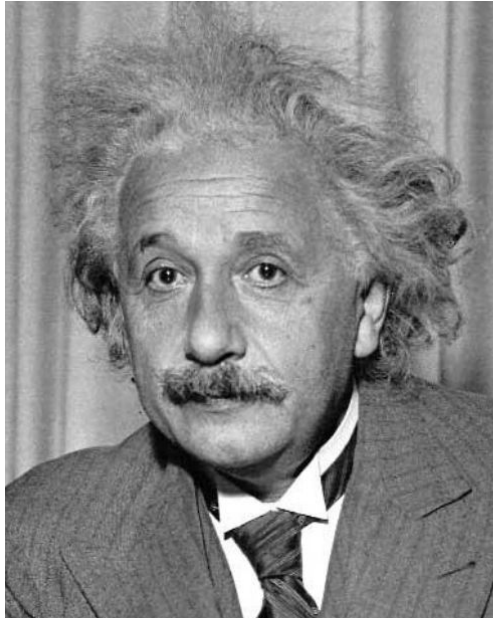


1	0	-1
2	0	-2
1	0	-1

Sobel



Sobel Filter (horizontaal)



1	2	1
0	0	0
-1	-2	-1

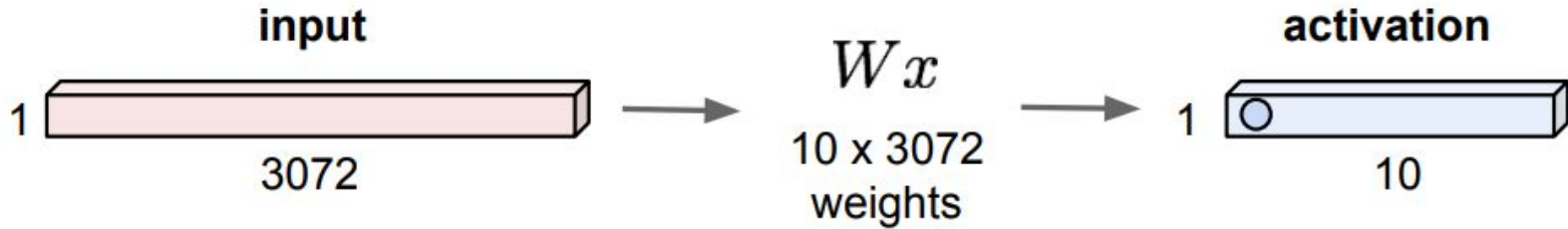
Sobel



Network Layers

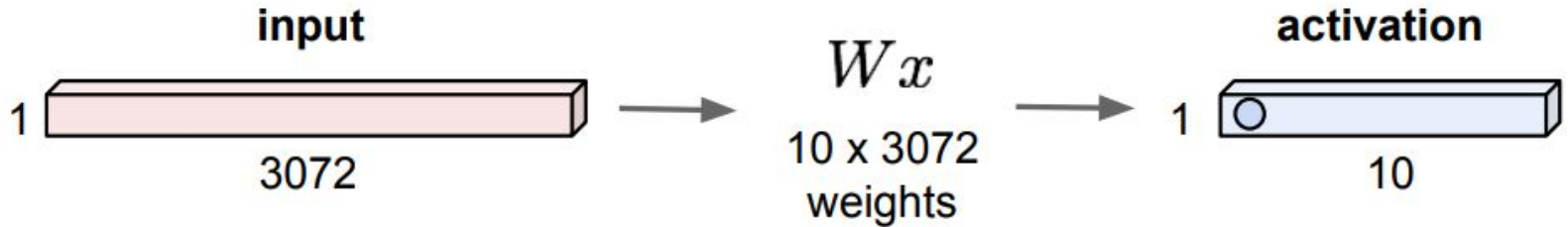
Fully Connected Layer

32x32x3 input afbeelding -> reshape naar 3072x1



Fully Connected Layer

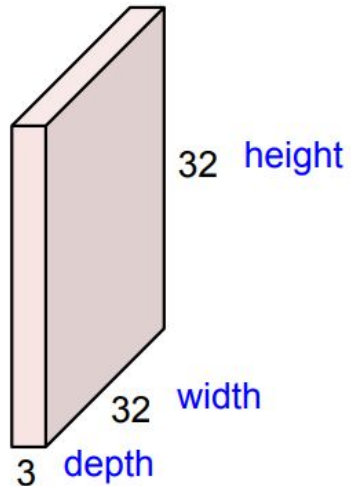
32x32x3 input afbeelding -> reshape naar 3072x1



Geeft 1 getal: dot product tussen een rij van W en de input (3072-dimensionaal dot product)

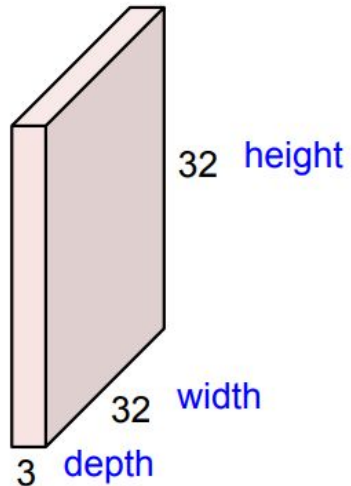
Convolutional Layer

- 32x32x3 input afbeelding (behoudt de spatial structure)

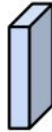


Convolutional Layer

- 32x32x3 input afbeelding (behoudt de spatial structure)
- Convolve het filter met de afbeelding (slide het filter over de afbeelding, bereken dot producten)



5x5x3 filter



Convolutional Layer

Wordt gebruikt om features in een afbeelding te matchen

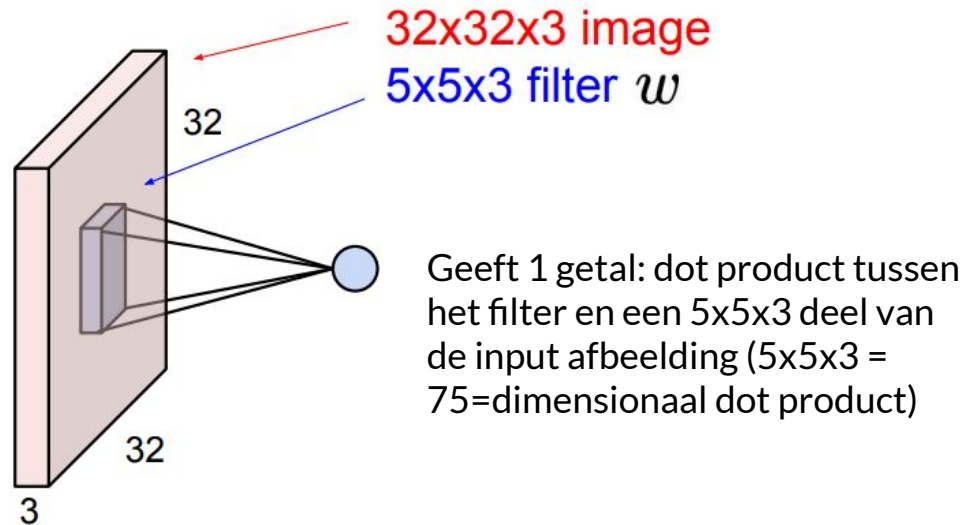
Filter

2	1	0
0	2	2
2	1	0

3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

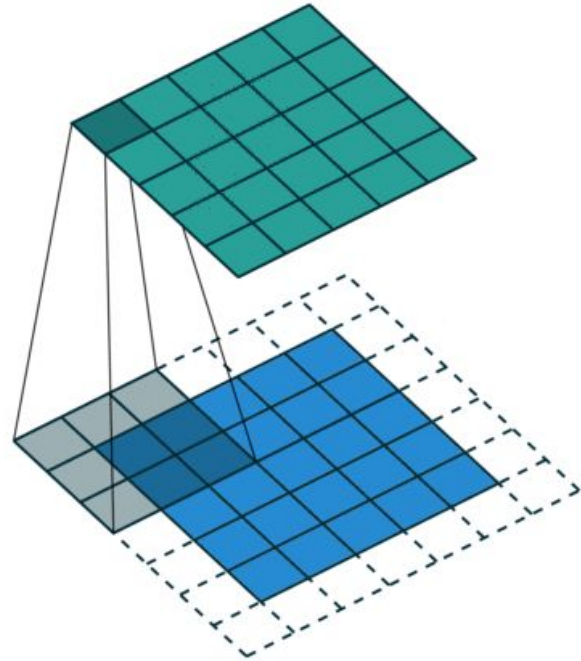
12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

Convolutional Layer



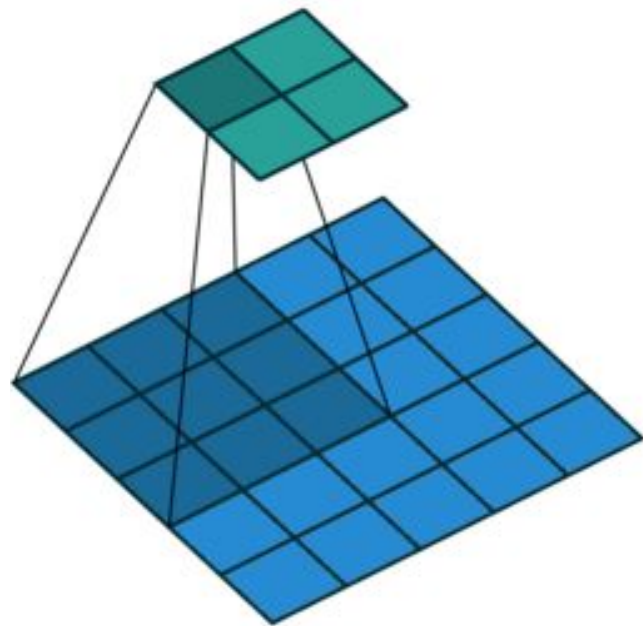
Padding

Zorgt dat ook de edges van de input worden meegenomen



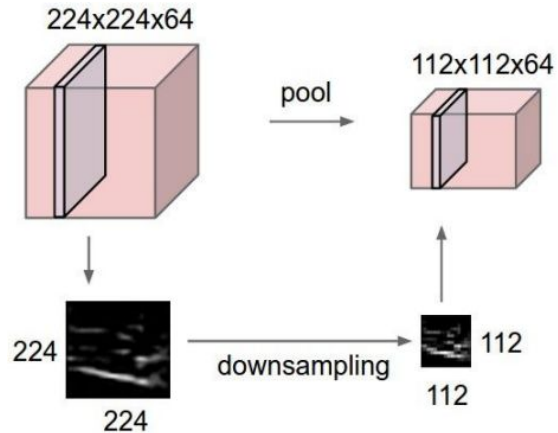
Stride

Verkleint de output size



Pooling Layer

- Verkleint de representaties
- Behoudt de belangrijkste informatie






Max Pooling

Neem het hoogste getal

Bijvoorbeeld: 2x2 max pooling filter

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4



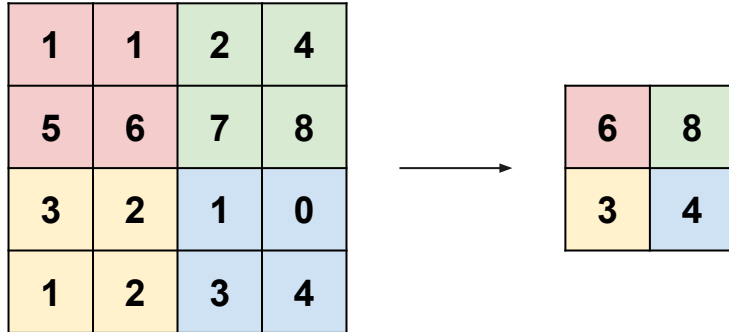
6	7	8
6	7	8
3	3	4



Max Pooling

Neem het hoogste getal

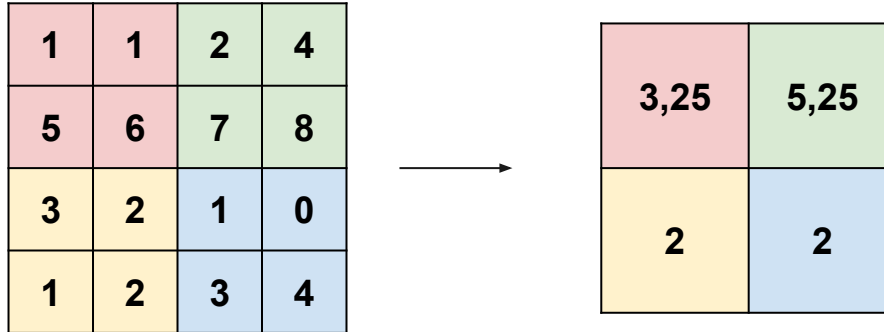
Bijvoorbeeld: 2x2 max pooling filter met stride 2



Average Pooling

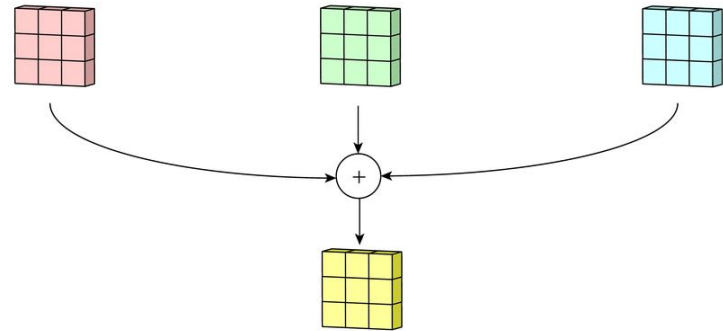
Neem het gemiddelde getal

Bijvoorbeeld: 2x2 avg pooling filter met stride 2



Filter v.s. kernel

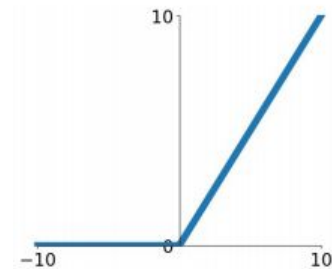
- 1D: filter = kernel
- n D: filter = verzameling van kernels
Eén kernel voor elk input channel
- Elk filter in een convolutional layer produceert één output channel



Activation Layer

ReLU (Rectified Linear Unit)

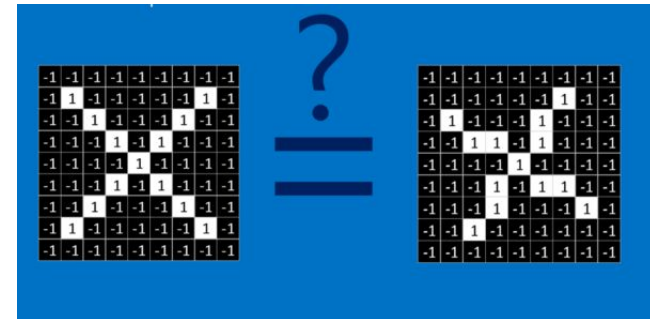
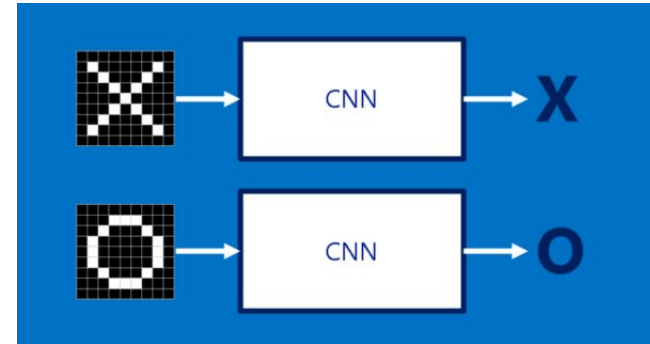
- Computationally efficient
- Snelle convergence
- Output is not zero-centered



Convolutional Neural Network

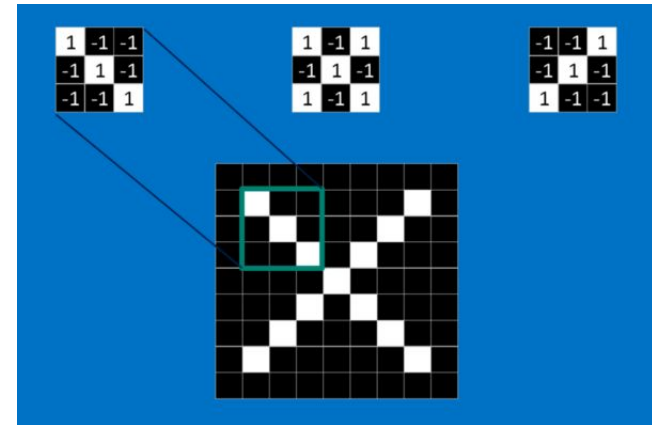
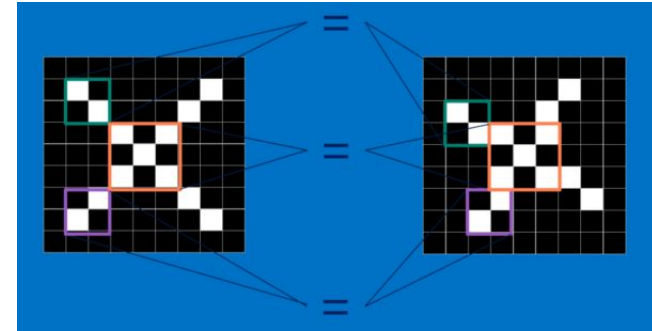
X'en en O's

- Naïeve aanpak:
 1. Bewaar een afbeelding van een X en een O
 2. Check welke het meest matcht met de input
- Een afbeelding is een 2D array van pixels
- Als de pixels niet matchen, matchen voor een computer de afbeeldingen niet
- We willen ook X'en en O's kunnen herkennen als ze zijn geshift, gekrompen, gedraaid of vervormd



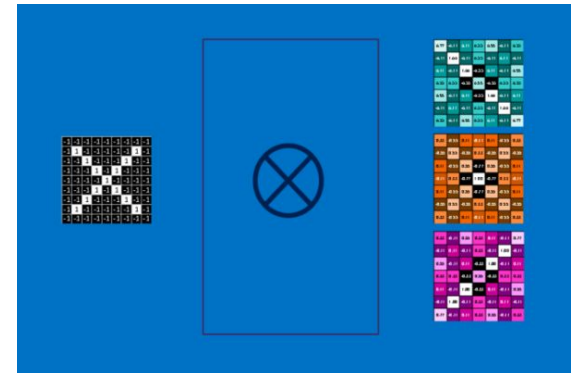
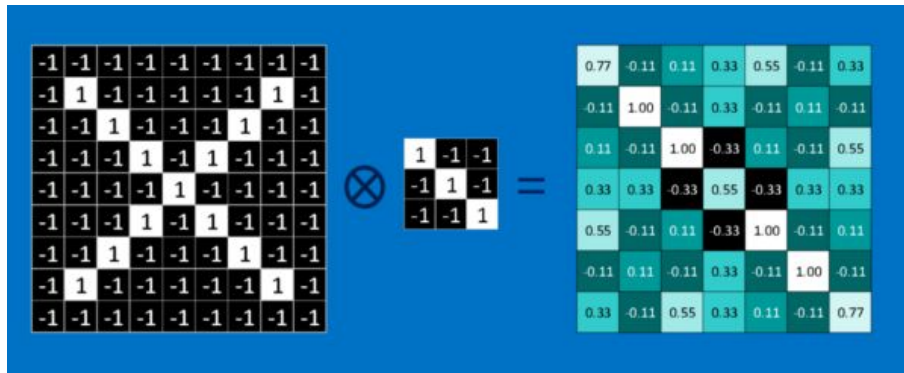
Features

- CNNs vergelijken afbeelding stukje voor stukje
- CNNs zoeken naar features
- Elke feature is als een mini afbeelding (een kleine 2D array)
- X'en: Features zijn diagonale lijnen en kruisingen

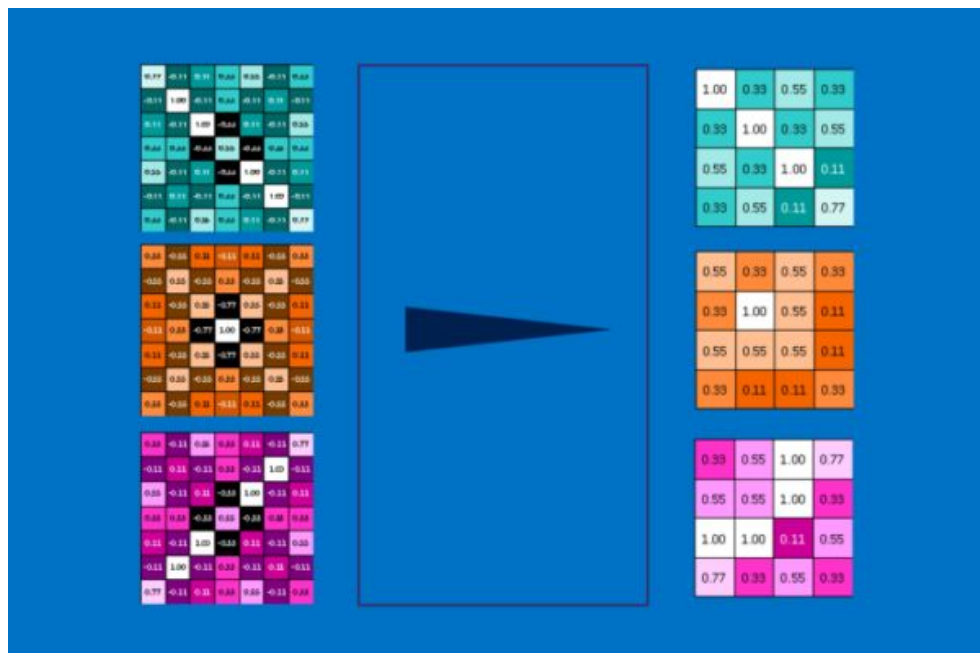


Convolutie

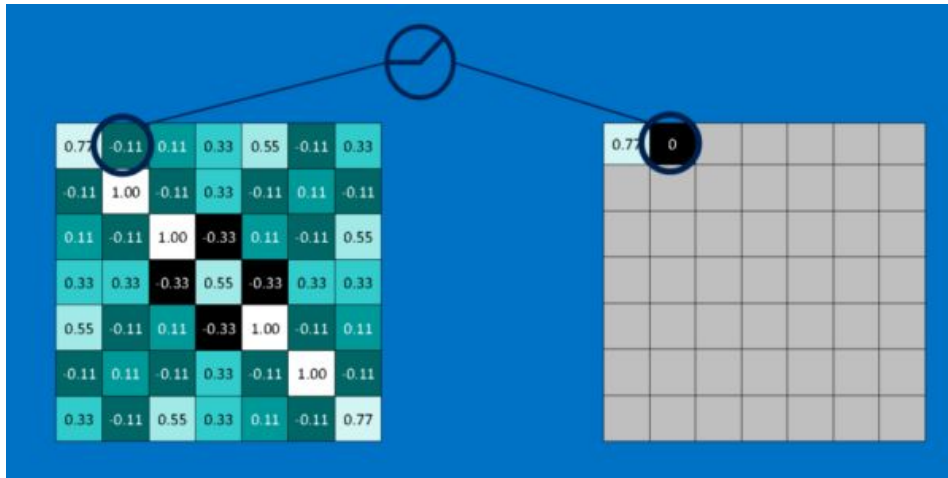
- De gehele afbeelding wordt doorzocht voor elke feature
- Geeft een set gefilterde afbeeldingen, één voor elke feature



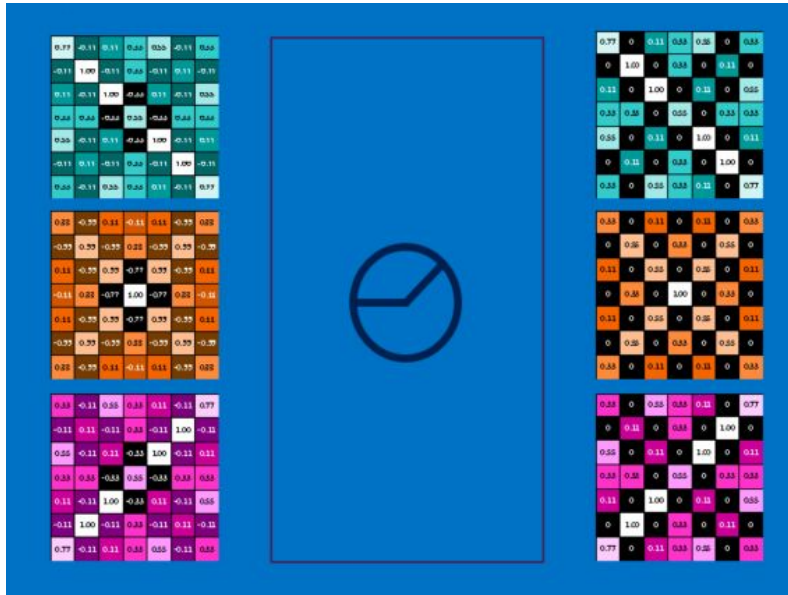
Pooling



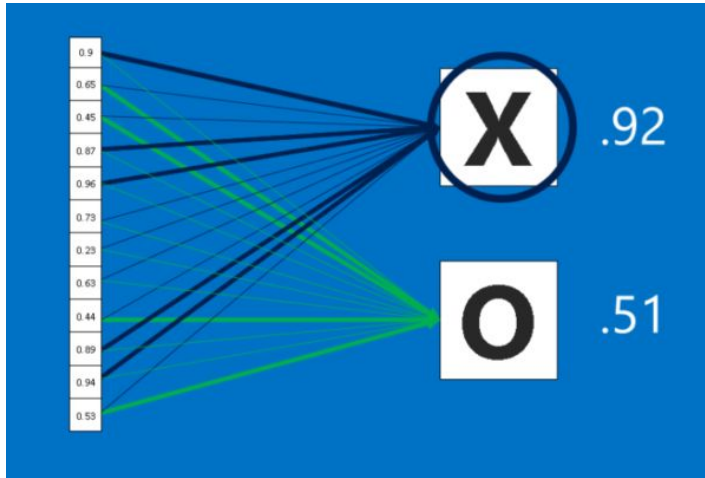
ReLU (Rectified Linear Units)



ReLU (Rectified Linear Units)

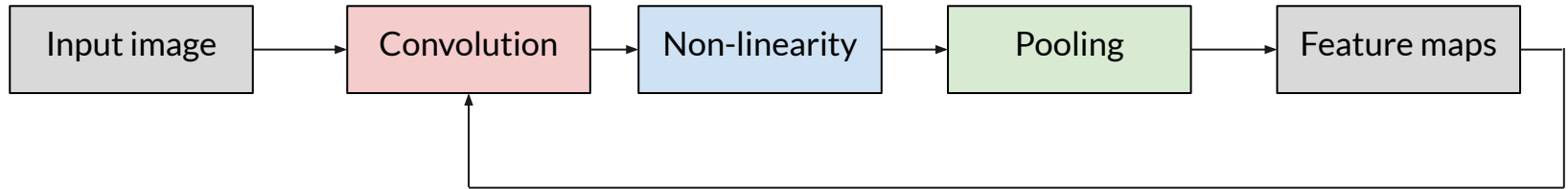


Fully Connected Layers





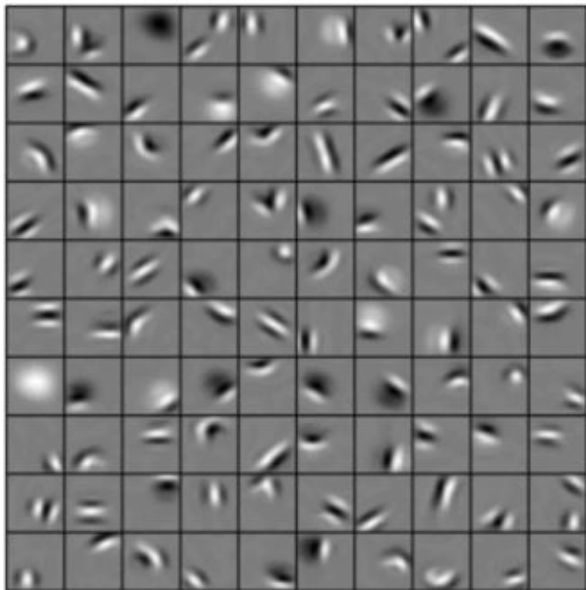
Model



Model



Higher v.s. lower layers





Backpropagation

- Gelabelde training data
- Initialisatie van pixels van features en weights van FC's
- Input image wordt voorspeld
- Foutheid van voorspelling vertelt hoe goed de features en weights zijn
- Features en weights worden aangepast om de error te verkleinen



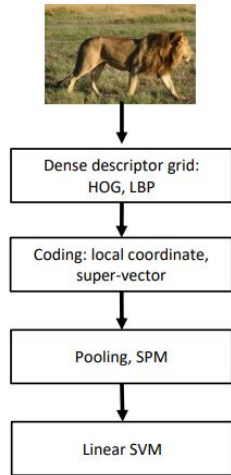
Waarom werken CNNs?

- Kernels combineren pixels in een kleine, lokale area om een output te vormen
De output feature ziet alleen de input features van een kleine, lokale area
- De kernel wordt over de gehele afbeelding gebruikt om outputs te produceren

- Architectuur: hoogte x breedte kleiner, meer channels
- Steeds complexere features

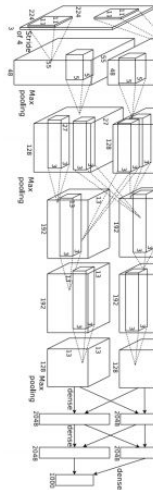
Evolutie van CNNs

2010



NEC-UIUC

2012



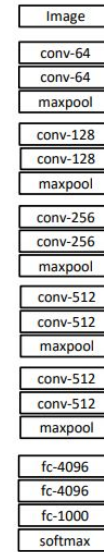
SuperVision

2014

● Pooling
● Convolution
● Softmax
● Other

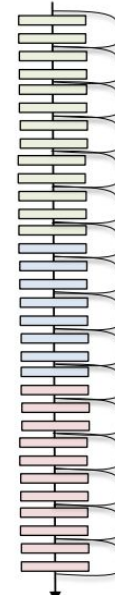


GoogLeNet



VGG

2015



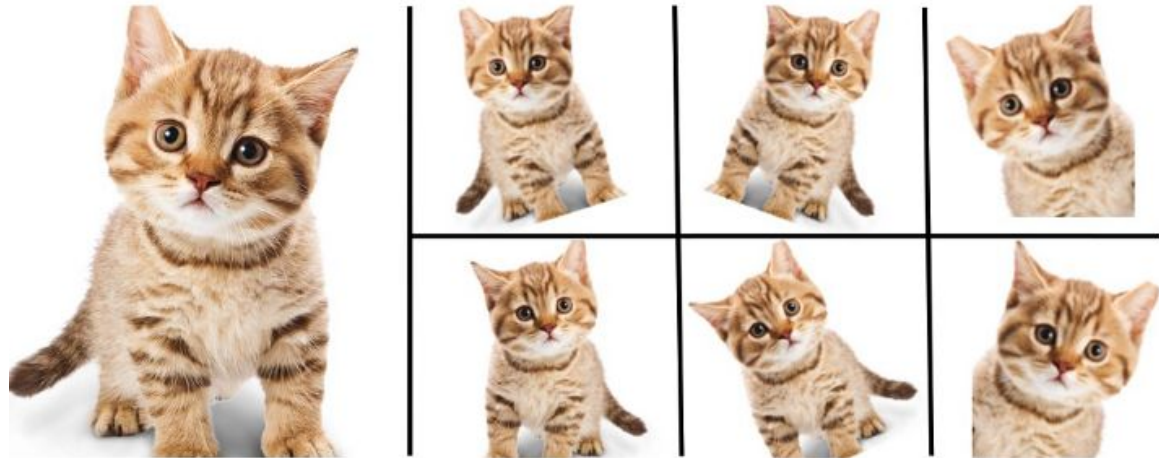
MSRA

Wat als je niet genoeg training data hebt?

Data Augmentation

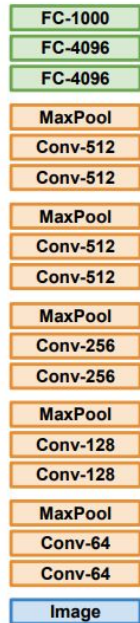
Transformaties toepassen op training afbeeldingen

- Resizing
- Cropping
- Flipping
- Rotation

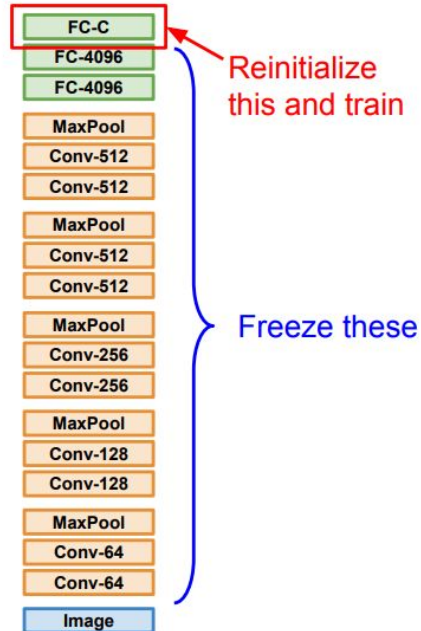


Transfer Learning

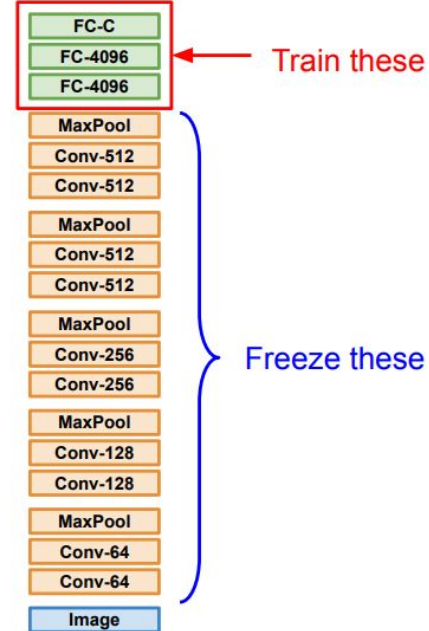
1. Train op ImageNet



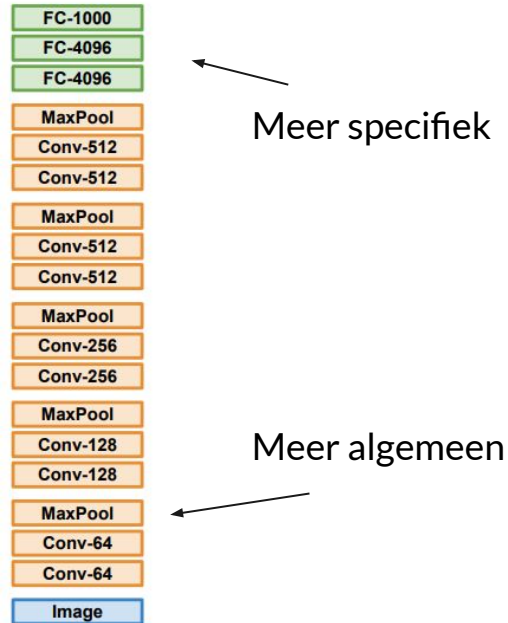
2. Kleine dataset



3. Grotere dataset

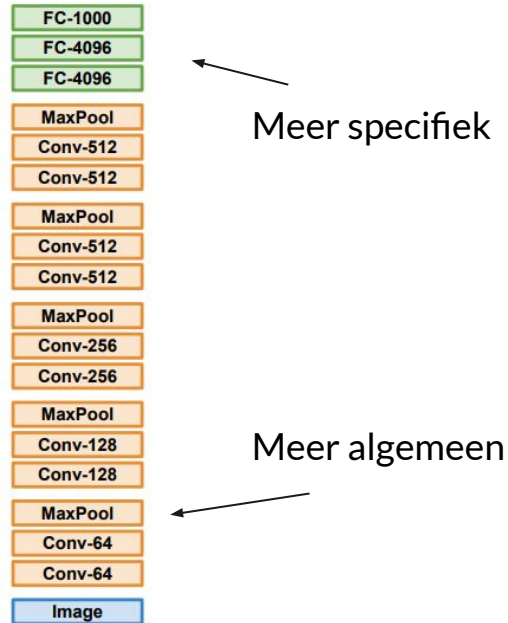


Transfer Learning



	Soortgelijke dataset	Andere dataset
Weinig data	?	?
Redelijk veel data	?	?

Transfer Learning



	Soortgelijke dataset	Andere dataset
Weinig data	Gebruik Linear Classifier in top layer	Probleem..
Redelijk veel data	Finetune een paar lagen	Finetune een grotere hoeveelheid lagen

Conclusie & Applicaties



Conclusie

- Computer Vision wordt gebruikt om foto's en video's automatisch te begrijpen
- CNNs werken goed omdat ze simpele en complexe features ontdekken
 - De lokale features moeten algemeen genoeg zijn om over de hele afbeelding gebruikt te kunnen worden
- CNNs worden steeds dieper
- Data augmentation en/of transfer learning nodig

Boodschappen doen



Boodschappen doen



- Bestaat al in Seattle
- Camera's op het plafond
- Computer vision bepaalt of een product is gepakt (en door wie)
- Mensen trainen de algoritmes bij door fouten op te sporen

Zelfrijdende Auto's



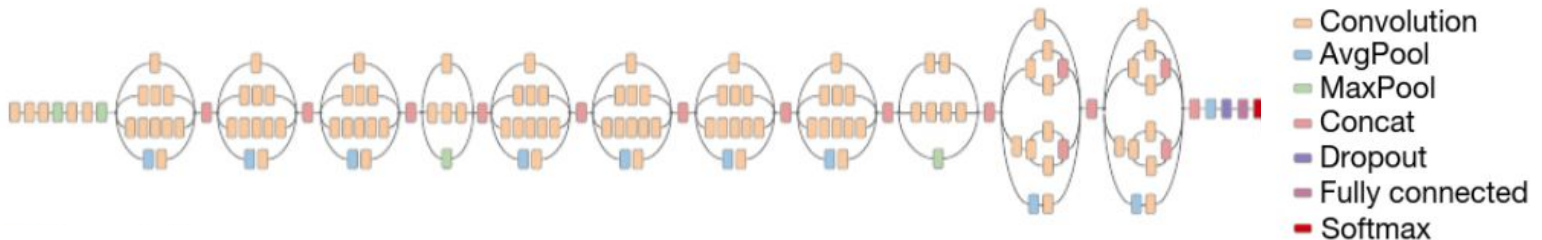
Zelfrijdende Auto's



- Gebruikt computer vision om objecten te detecteren
- Handsignalen van fietsers detecteren

Gezondheidszorg

- Kanker detecteren in scans
- Deep CNN
 - Pretrained op ImageNet (1,28 miljoen images van 1,000 objecten)
 - Fine-tuned op eigen dataset (129,450 images)
- 21 dermatologen
- CNN detecteert even goed als dermatologen



Dermatologist-level classification of skin cancer with deep neural networks

Andre Esteva^{1,*}, Brett Kuprel^{1,*}, Roberto A. Novoa^{2,3}, Justin Ko², Susan M. Swetter^{2,4}, Helen M. Blau⁵ & Sebastian Thrun⁶

Skin cancer, the most common human malignancy¹⁻³, is primarily diagnosed visually, beginning with an initial clinical screening and followed potentially by dermoscopic analysis, a biopsy and histopathological examination. Automated classification of skin lesions using images is a challenging task owing to the fine-grained variability in the appearance of skin lesions. Deep convolutional

images (for example, smartphone images) exhibit variability in factors such as zoom, angle and lighting, making classification substantially more challenging^{23,24}. We overcome this challenge by using a data-driven approach—1.41 million pre-training and training images make classification robust to photographic variability. Many previous techniques require extensive preprocessing, lesion segmentation and



Landbouw

- Drones maken afbeeldingen van de bodem, die worden geanalyseerd om te bepalen of de grond vruchtbaar is of dat er moet worden geoogst
- Stenen moeten weggehaald worden van gewassen voordat er gezaaid kan worden
 - Kunnen automatisch verwijderd worden met CV en robots

Assignment 1

<https://git.io/JvB50>

Introduction to Generative Models

Supervised v.s. Unsupervised Learning

Supervised Learning

- Data: (x, y)
- Doel: Leer een functie die $x \rightarrow y$ mapt
- Voorbeelden: classification, regression, object detection, semantic segmentation



→ Cat



DOG, DOG, CAT

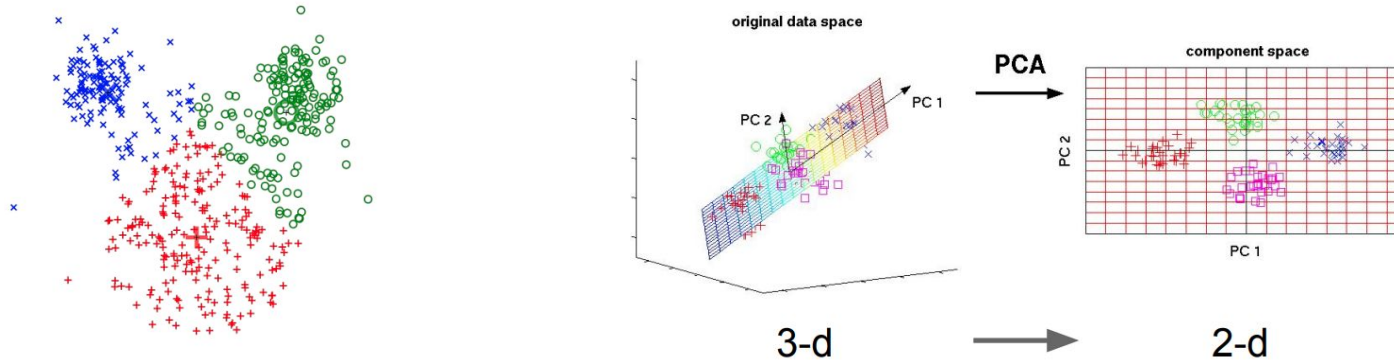


GRASS, CAT,
TREE, SKY

Supervised v.s. Unsupervised Learning

Unsupervised Learning

- Data: (x)
- Doel: Leer een onderliggende, verborgen structuur van de data
- Voorbeelden: clustering, dimensionality reduction



Generative Models

- Genereer nieuwe samples van dezelfde distributie als de training data
- Leer $p_{\text{model}}(x)$, vergelijkbaar met $p_{\text{data}}(x)$



Training data $\sim p_{\text{data}}(x)$



Generated samples $\sim p_{\text{model}}(x)$

Generative Models

- Genereer nieuwe samples van dezelfde distributie als de training data
- Leer $p_{\text{model}}(x)$, vergelijkbaar met $p_{\text{data}}(x)$



Training data $\sim p_{\text{data}}(x)$



Generated samples $\sim p_{\text{model}}(x)$

- Density estimation: schatting van een onobserveerbare, onderliggende probability density function (kansdichtheid) maken op basis van geobserveerde data
 - a. Explicit density estimation: Definieer en los expliciet op voor $p_{\text{model}}(x)$
 - b. Implicit density estimation: Leer een model dat kan samplen van $p_{\text{model}}(x)$ zonder deze expliciet te definiëren

Generative Models

- Produceren realistische nieuwe samples voor:
 - Kunst
 - Gezichten
 - Image super-resolution: Hoge resolutie versies genereren van input afbeeldingen (medical imaging)



2014



2015

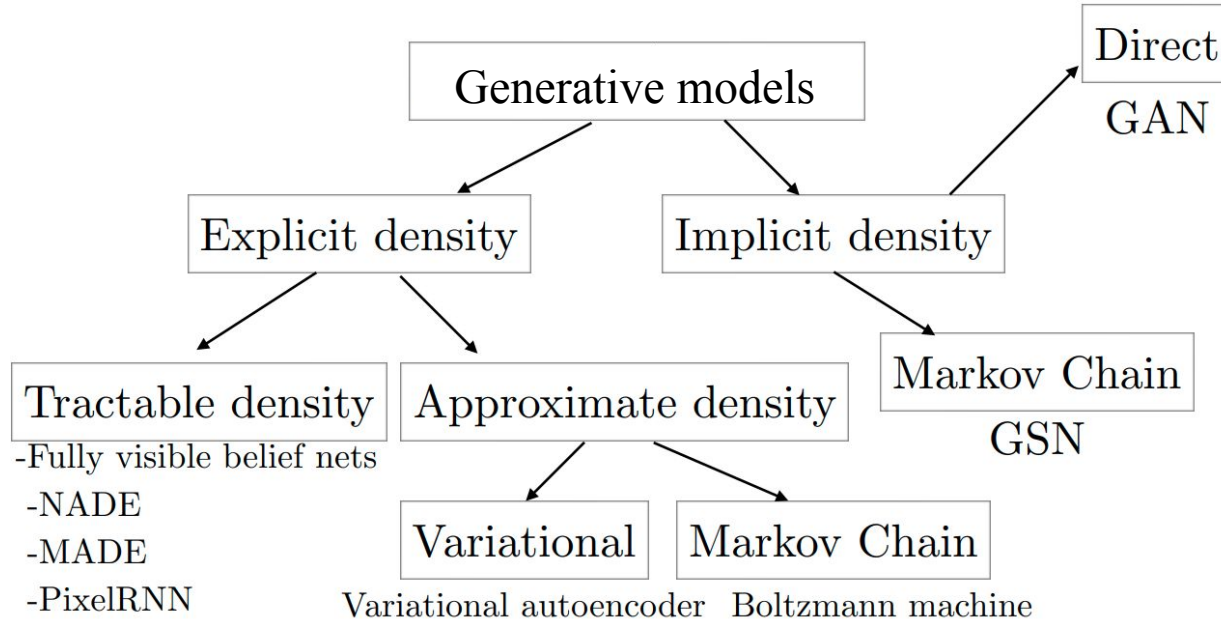


2017

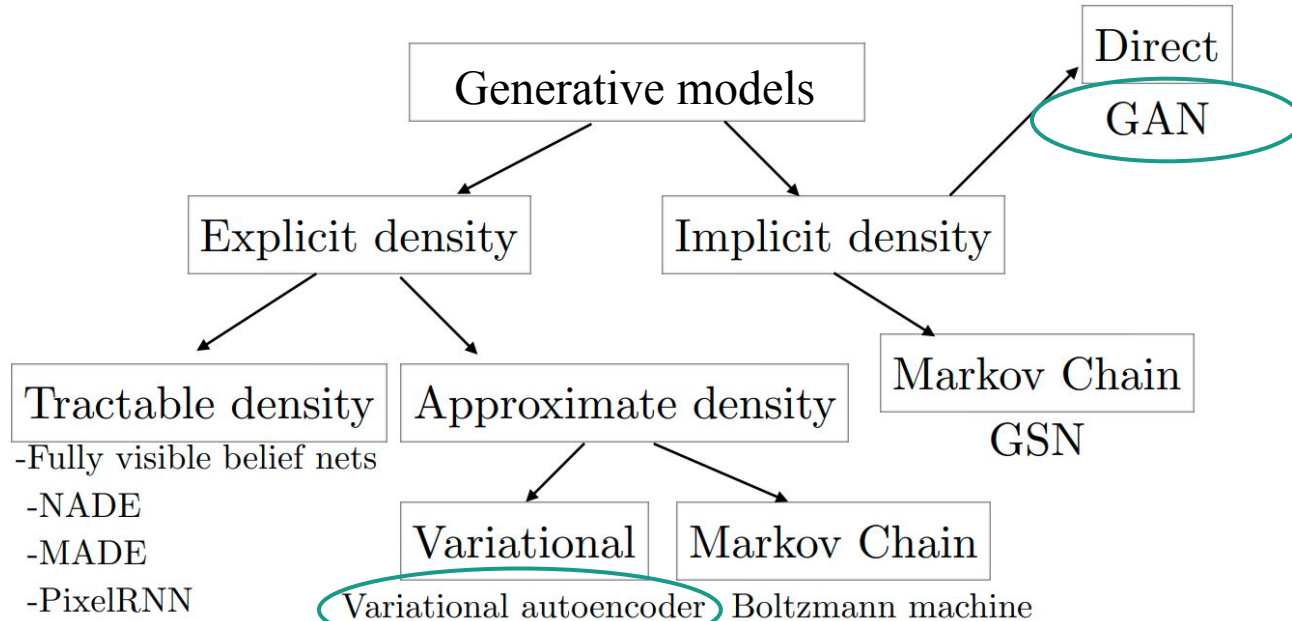


2017

Generative Models



Generative Models

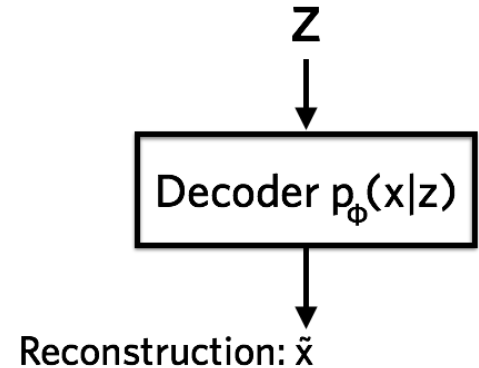
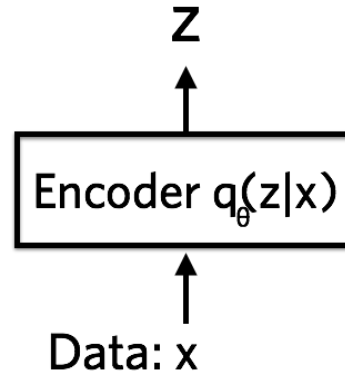


Variational Autoencoders

Neural Network Perspectief

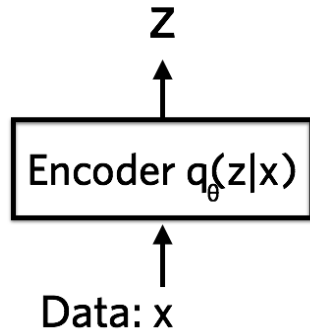
Onderdelen:

1. Encoder
Compresst de data in een latent space Z
2. Decoder
Reconstrueert de data met de hidden representation
3. Loss function



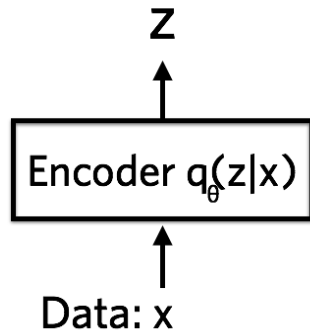
Encoder

- Neural Network
- Input: datapunt x
- Output: hidden representation z
- Heeft weights and biases θ



Encoder

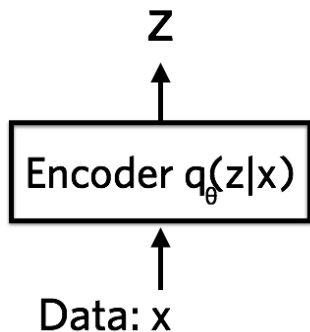
- Neural Network
- Input: datapunt x
- Output: hidden representation z
- Heeft weights and biases θ



- Input x : 28x28 afbeelding van een handgeschreven getal
- Encoder encodeert data (784-dimensionaal) in een hidden representation space z (kleiner dan 784D)
- Encoder moet een efficiënte compressie van de data leren

Encoder

- Neural Network
- Input: datapunt x
- Output: hidden representation z
- Heeft weights and biases θ

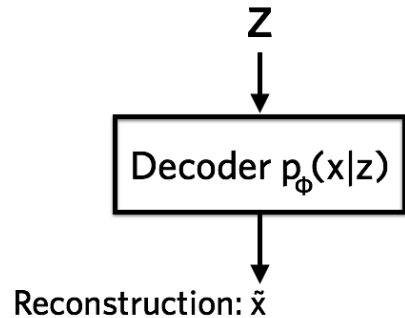


- Input x : 28x28 afbeelding van een handgeschreven getal
- Encoder encodeert data (784-dimensionaal) in een hidden representation space z (kleiner dan 784D)
- Encoder moet een efficiënte compressie van de data leren

- Encoder: $q_{\theta}(z|x)$
- Hidden space z is stochastisch: encoder geeft parameters voor $q_{\theta}(z|x)$ (normaalverdeling)
- Kunnen van deze distributie samplen om noisy values te krijgen van de representaties z

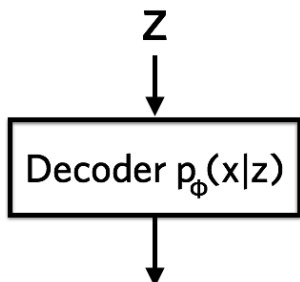
Decoder

- Neural Network
- Input: representatie z
- Output: parameters voor de probability distribution van de data
- Heeft weights en biases ϕ



Decoder

- Neural Network
- Input: representatie z
- Output: parameters voor de probability distribution van de data
- Heeft weights en biases ϕ

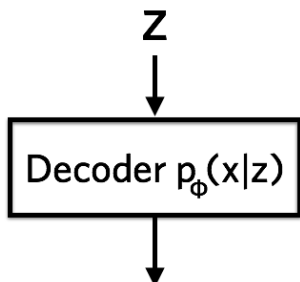


Reconstruction: \tilde{x}

- Elke pixel op de afbeelding is een 0 of 1
- Bernoulli distributie
- Decoder krijgt de latent representation van een getal z als input
- Geeft 784 Bernoulli parameters als output, één voor elke pixel
- Decoder decodeert de nummers in z naar 784 getallen tussen 0 en 1

Decoder

- Neural Network
- Input: representatie z
- Output: parameters voor de probability distribution van de data
- Heeft weights en biases ϕ



Reconstruction: \tilde{x}

- Elke pixel op de afbeelding is een 0 of 1
- Bernoulli distributie
- Decoder krijgt de latent representation van een getal z als input
- Geeft 784 Bernoulli parameters als output, één voor elke pixel
- Decoder decodeert de nummers in z naar 784 getallen tussen 0 en 1
- Decoder: $p_{\phi}(x|z)$
- Informatie van de oorspronkelijke 784D vector kan niet perfect overgebracht worden
- Decoder heeft alleen toegang tot een samenvatting van de informatie (kleiner dan 784D)



Loss function

Encoder: $q_{\theta}(z|x)$
Decoder: $p_{\phi}(x|z)$

Hoeveel informatie gaat verloren?

- Meet met negative log likelihood $p_{\phi}(x|z)$
- Vertelt hoe effectief de decoder heeft geleerd een input image x te reconstrueren gegeven zijn latent representation z

$$\ell_i(\theta, \phi) = -\mathbb{E}_{z \sim q_{\theta}(z|x_i)} [\log p_{\phi}(x_i|z)] + \text{KL}(q_{\theta}(z|x_i) \parallel p(z))$$



Loss function

Encoder: $q_{\theta}(z|x)$
Decoder: $p_{\phi}(x|z)$

Hoeveel informatie gaat verloren?

- Meet met negative log likelihood $p_{\phi}(x|z)$
- Vertelt hoe effectief de decoder heeft geleerd een input image x te reconstrueren gegeven zijn latent representation z
- Er zijn geen globale representaties die worden gedeeld door alle datapunten

Loss functie kan worden ontbonden in termen die alleen van één datapunt (ℓ_i) afhangen

Som over alle datapunten

$$\ell_i(\theta, \phi) = -\mathbb{E}_{z \sim q_{\theta}(z|x_i)} [\log p_{\phi}(x_i|z)] + \text{KL}(q_{\theta}(z|x_i) \parallel p(z))$$

$$\mathcal{L}(\theta, \phi) = \sum_{i=1}^N \ell_i$$

Loss function

Encoder: $q_{\theta}(z|x)$
Decoder: $p_{\varphi}(x|z)$

1. Reconstruction loss

- a. Expectation over encoder's distributie over de representaties
- b. Leert de decoder de data te reconstrueren

2. Regularizer

- a. Kullback-Leibler divergence tussen encoder's distribution $q_{\theta}(z|x)$ en prior $p(z)$
- b. Meet hoeveel informatie verloren gaat wanneer we q gebruiken om p te representeren

$$\ell_i(\theta, \varphi) = -\mathbb{E}_{z \sim q_{\theta}(z|x_i)} [\log p_{\varphi}(x_i|z)] + \text{KL}(q_{\theta}(z|x_i) \parallel p(z))$$

$$\mathcal{L}(\theta, \varphi) = \sum_{i=1}^N \ell_i$$

Loss function

Encoder: $q_{\theta}(z|x)$
Decoder: $p_{\phi}(x|z)$

1. Reconstruction loss

- a. Expectation over encoder's distributie over de representaties
- b. Leert de decoder de data te reconstrueren

2. Regularizer

- a. Kullback-Leibler divergence tussen encoder's distribution $q_{\theta}(z|x)$ en prior $p(z)$
- b. Meet hoeveel informatie verloren gaat wanneer we q gebruiken om p te representeren
- c. Als de encoder representaties geeft die anders zijn dan de standaardnormale verdeling, krijgt hij een penalty
- d. Zorgt dat representaties van z van elk getal divers zijn

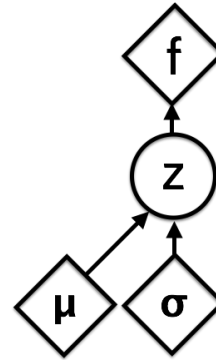
$p(z)$: standaardnormale verdeling
 $p(z) = \mathcal{N}(0, 1)$

$$\ell_i(\theta, \phi) = -\mathbb{E}_{z \sim q_{\theta}(z|x_i)} [\log p_{\phi}(x_i|z)] + \text{KL}(q_{\theta}(z|x_i) \parallel p(z))$$

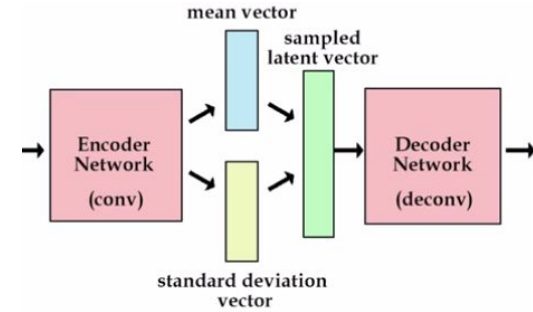
$$\mathcal{L}(\theta, \phi) = \sum_{i=1}^N \ell_i$$

Reparameterization Trick

- Decoder input: sampled representation z
- Samples hebben geen gradients
- Backpropagation



Original

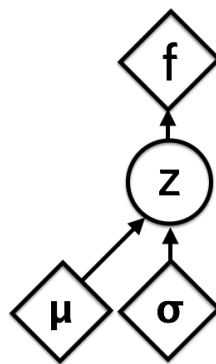
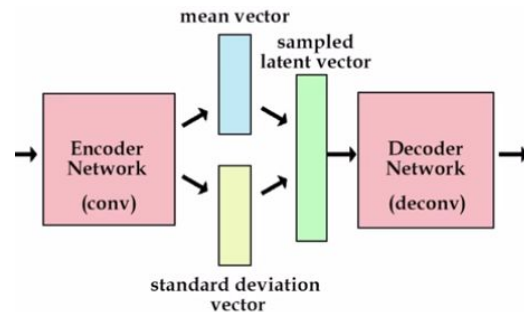


Reparameterization Trick

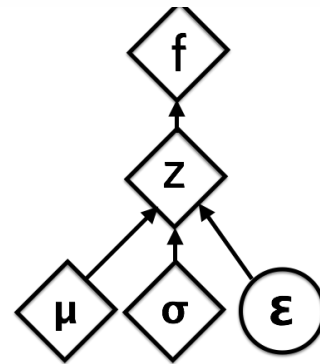
- Decoder input: sampled representation z
- Samples hebben geen gradients
- Backpropagation
- Verplaats sampling step
- Normaalverdeelde variabele met gemiddelde μ , standaarddeviatie σ

$$z = \mu + \sigma \cdot \epsilon$$

waar $\epsilon \sim \mathcal{N}(0, 1)$



Original



Reparametrized



Probability Model Perspectief

Onderdelen:

- Probability model van data x
- Latent variables z

Joint probability van het model:

$$p(x, z) = p(x|z) p(z)$$

Probability Model Perspectief

Onderdelen:

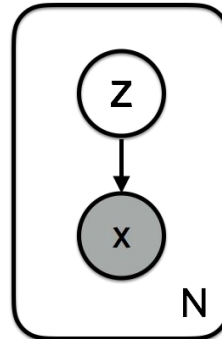
- Probability model van data x
- Latent variables z

Joint probability van het model:

$$p(x, z) = p(x|z) p(z)$$

Voor elk datapunt i :

1. Trek latent variables z uit prior $p(z)$: $z_i \sim p(z)$
2. Trek datapunt $x_i \sim p(x|z)$



Probability Model Perspectief

Onderdelen:

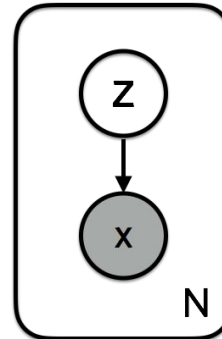
- Probability model van data x
- Latent variables z

Joint probability van het model:

$$p(x, z) = p(x|z) p(z)$$

Voor elk datapunt i :

1. Trek latent variables z uit prior $p(z): z_i \sim p(z)$
2. Trek datapunt $x_i \sim p(x|z)$



Likelihood $p(x|z)$
hangt af van de
latent variables z

$\mathcal{N}(0, 1)$

Probability Model Perspectief

Onderdelen:

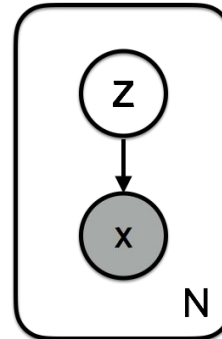
- Probability model van data x
- Latent variables z

Joint probability van het model:

$$p(x, z) = p(x|z) p(z)$$

Voor elk datapunt i :

1. Trek latent variables z uit prior $p(z)$: $z_i \sim p(z)$
2. Trek datapunt $x_i \sim p(x|z)$



Likelihood $p(x|z)$
hangt af van de
latent variables z

Bernoulli
verdeling

$\mathcal{N}(0, 1)$

Nieuwe voorspellingen maken

Doel:

1. Goede waarden afleiden van de latent variables gegeven de geobserveerde data
2. Posterior distribution $p(z|x)$ berekenen

Bayes Theorem:

$$\text{Posterior } \left\{ p(z|x) = \frac{\overbrace{p(x|z)}^{\text{Likelihood}} \overbrace{p(z)}^{\text{Prior}}}{\underbrace{p(x)}_{\text{Evidence}}} = \frac{p(x|z)p(z)}{\int p(x|z)p(z)dz} \right.$$

Integraal kost exponentiële
hoeveelheid tijd om te berekenen

Nieuwe voorspellingen maken

Doel:

1. Goede waarden afleiden van de latent variables gegeven de geobserveerde data
2. Posterior distribution $p(z|x)$ berekenen

Benadering!

Bayes Theorem:

$$\text{Posterior } \left\{ p(z|x) = \frac{\overbrace{p(x|z)p(z)}^{\text{Likelihood Prior}}}{\underbrace{p(x)}_{\text{Evidence}}} = \frac{p(x|z)p(z)}{\int p(x|z)p(z)dz} \right.$$

Integraal kost exponentiële
hoeveelheid tijd om te berekenen



Variational Inference

- Benader de true posterior $p(z|x)$ met een variational posterior $q_\lambda(z|x)$
- λ indiceert de gekozen distributie
 - Bijvoorbeeld: Normaalverdeling
 - $\lambda_{xi} = (\mu_{xi}, \sigma_{xi}^2)$
- Meet hoe goed de variational posterior de true posterior benadert
 - Kullback-Leibler divergence
 - Meet hoeveel informatie verloren gaat wanneer we q gebruiken om p te benaderen

$$\text{KL}(q_\lambda(z|x) \parallel p(z|x)) = \mathbb{E}_q[\log q_\lambda(z|x)] - \mathbb{E}_q[\log p(x, z)] + \log p(x)$$



Variational Inference

- Benader de true posterior $p(z|x)$ met een variational posterior $q_\lambda(z|x)$
- λ indiceert de gekozen distributie
 - Bijvoorbeeld: Normaalverdeling
 - $\lambda_{xi} = (\mu_{xi}, \sigma_{xi}^2)$
- Meet hoe goed de variational posterior de true posterior benadert
 - Kullback-Leibler divergence
 - Meet hoeveel informatie verloren gaat wanneer we q gebruiken om p te benaderen

$$\text{KL}(q_\lambda(z|x) \parallel p(z|x)) = \mathbb{E}_q[\log q_\lambda(z|x)] - \mathbb{E}_q[\log p(x, z)] + \log p(x)$$

Doel: vind de variational parameters λ die de KL-divergence minimaliseren

$$q_\lambda^* = \arg \min_\lambda \text{KL}(q_\lambda(z|x) \parallel p(z|x))$$



Variational Inference

$$q_{\lambda}^* = \arg \min_{\lambda} \text{KL}(q_{\lambda}(z|x) || p(z|x))$$

$$\text{KL}(q_{\lambda}(z|x) || p(z|x)) = \mathbb{E}_q[\log q_{\lambda}(z|x)] - \mathbb{E}_q[\log p(x, z)] + \log p(x)$$

- Intractable door $p(x)$
- Nieuwe functie:

$$\text{ELBO}(\lambda) = \mathbb{E}_q[\log p(x, z)] - \mathbb{E}_q[\log q_{\lambda}(z|x)]$$



Evidence Lower BOund

$$\text{ELBO}(\lambda) = \mathbb{E}_q[\log p(x, z)] - \mathbb{E}_q[\log q_\lambda(z|x)]$$

$$\text{KL}(q_\lambda(z|x) || p(z|x)) = -\mathbb{E}_q[\log p(x, z)] + \mathbb{E}_q[\log q_\lambda(z|x)] + \log p(x)$$

$$\log p(x) = \text{KL}(q_\lambda(z|x) || p(z|x)) + \mathbb{E}_q[\log p(x, z)] - \mathbb{E}_q[\log q_\lambda(z|x)]$$

$$\log p(x) = \text{KL}(q_\lambda(z|x) || p(z|x)) + \text{ELBO}(\lambda)$$



Evidence Lower BOund

$$\text{ELBO}(\lambda) = \mathbb{E}_q[\log p(x, z)] - \mathbb{E}_q[\log q_\lambda(z|x)]$$

$$\text{KL}(q_\lambda(z|x) \parallel p(z|x)) = -\mathbb{E}_q[\log p(x, z)] + \mathbb{E}_q[\log q_\lambda(z|x)] + \log p(x)$$

$$\log p(x) = \text{KL}(q_\lambda(z|x) \parallel p(z|x)) + \mathbb{E}_q[\log p(x, z)] - \mathbb{E}_q[\log q_\lambda(z|x)]$$

$$\log p(x) = \text{KL}(q_\lambda(z|x) \parallel p(z|x)) + \text{ELBO}(\lambda)$$

- KL eigenschap: $\text{KL} \geq 0$
- KL minimaliseren == ELBO maximaliseren tot constante: $\log p(x)$



Evidence Lower BOund

$$\begin{aligned}\text{ELBO}_i(\lambda) &= \mathbb{E}_q[\log p(x_i, z)] - \mathbb{E}_q[\log q_\lambda(z|x_i)] \\ &= \mathbb{E}_q[\log(p(x_i|z) p(z))] - \mathbb{E}_q[\log q_\lambda(z|x_i)] \\ &= \mathbb{E}_q[\log p(x_i|z)] + \mathbb{E}_q[\log p(z)] - \mathbb{E}_q[\log q_\lambda(z|x_i)]\end{aligned}$$

$$\text{KL}(q_\lambda(z|x_i) || p(z)) = \mathbb{E}_q\left[\log \frac{q_\lambda(z|x_i)}{p(z)}\right]$$



Evidence Lower BOund

$$\begin{aligned}\text{ELBO}_i(\lambda) &= \mathbb{E}_q[\log p(x_i, z)] - \mathbb{E}_q[\log q_\lambda(z|x_i)] \\ &= \mathbb{E}_q[\log(p(x_i|z) p(z))] - \mathbb{E}_q[\log q_\lambda(z|x_i)] \\ &= \mathbb{E}_q[\log p(x_i|z)] + \mathbb{E}_q[\log p(z)] - \mathbb{E}_q[\log q_\lambda(z|x_i)]\end{aligned}$$

$$\begin{aligned}-\text{ELBO}_i(\lambda) &= -\mathbb{E}_q[\log p(x_i|z)] - \mathbb{E}_q[\log p(z)] + \mathbb{E}_q[\log q_\lambda(z|x_i)] \\ &= -\mathbb{E}_q[\log p(x_i|z)] + \mathbb{E}_q[\log q_\lambda(z|x_i)] - \mathbb{E}_q[\log p(z)] \\ &= -\mathbb{E}_q[\log p(x_i|z)] + \mathbb{E}_q[\log \frac{q_\lambda(z|x_i)}{p(z)}]\end{aligned}$$

$$\text{KL}(q_\lambda(z|x_i) || p(z)) = \mathbb{E}_q[\log \frac{q_\lambda(z|x_i)}{p(z)}]$$



Evidence Lower BOund

$$\begin{aligned}\text{ELBO}_i(\lambda) &= \mathbb{E}_q[\log p(x_i, z)] - \mathbb{E}_q[\log q_\lambda(z|x_i)] \\ &= \mathbb{E}_q[\log(p(x_i|z) p(z))] - \mathbb{E}_q[\log q_\lambda(z|x_i)] \\ &= \mathbb{E}_q[\log p(x_i|z)] + \mathbb{E}_q[\log p(z)] - \mathbb{E}_q[\log q_\lambda(z|x_i)]\end{aligned}$$

$$\begin{aligned}-\text{ELBO}_i(\lambda) &= -\mathbb{E}_q[\log p(x_i|z)] - \mathbb{E}_q[\log p(z)] + \mathbb{E}_q[\log q_\lambda(z|x_i)] \\ &= -\mathbb{E}_q[\log p(x_i|z)] + \mathbb{E}_q[\log q_\lambda(z|x_i)] - \mathbb{E}_q[\log p(z)] \\ &= -\mathbb{E}_q[\log p(x_i|z)] + \mathbb{E}_q[\log \frac{q_\lambda(z|x_i)}{p(z)}] \\ &= -\mathbb{E}_q[\log p(x_i|z)] + \text{KL}(q_\lambda(z|x_i) || p(z))\end{aligned}$$

$$\text{ELBO}_i(\lambda) = \mathbb{E}_q[\log p(x_i|z)] - \text{KL}(q_\lambda(z|x_i) || p(z))$$

$$\text{KL}(q_\lambda(z|x_i) || p(z)) = \mathbb{E}_q[\log \frac{q_\lambda(z|x_i)}{p(z)}]$$

Evidence Lower BOund

$$\begin{aligned}\text{ELBO}_i(\lambda) &= \mathbb{E}_q[\log p(x_i, z)] - \mathbb{E}_q[\log q_\lambda(z|x_i)] \\ &= \mathbb{E}_q[\log(p(x_i|z) p(z))] - \mathbb{E}_q[\log q_\lambda(z|x_i)] \\ &= \mathbb{E}_q[\log p(x_i|z)] + \mathbb{E}_q[\log p(z)] - \mathbb{E}_q[\log q_\lambda(z|x_i)]\end{aligned}$$

$$\begin{aligned}-\text{ELBO}_i(\lambda) &= -\mathbb{E}_q[\log p(x_i|z)] - \mathbb{E}_q[\log p(z)] + \mathbb{E}_q[\log q_\lambda(z|x_i)] \\ &= -\mathbb{E}_q[\log p(x_i|z)] + \mathbb{E}_q[\log q_\lambda(z|x_i)] - \mathbb{E}_q[\log p(z)] \\ &= -\mathbb{E}_q[\log p(x_i|z)] + \mathbb{E}_q[\log \frac{q_\lambda(z|x_i)}{p(z)}] \\ &= -\mathbb{E}_q[\log p(x_i|z)] + \text{KL}(q_\lambda(z|x_i) \parallel p(z))\end{aligned}$$

$$\begin{aligned}\text{ELBO}_i(\lambda) &= \mathbb{E}_q[\log p(x_i|z)] - \text{KL}(q_\lambda(z|x_i) \parallel p(z)) \\ \text{ELBO}(\lambda) &= \sum_{i=1}^N \mathbb{E}_q[\log p(x_i|z)] - \text{KL}(q_\lambda(z|x_i) \parallel p(z))\end{aligned}$$

$$\text{KL}(q_\lambda(z|x_i) \parallel p(z)) = \mathbb{E}_q[\log \frac{q_\lambda(z|x_i)}{p(z)}]$$

Geen enkel datapunt deelt zijn latent z met de latent variable van een ander datapunt



Perspectieven verbinden

- Approximate posterior $q_{\theta}(z|x, \lambda)$ wordt geparameteriseerd met een inference network (encoder)
 - Input: data x
 - Output: parameters λ
- Likelihood $p(x|z)$ wordt geparameteriseerd met een generative network (decoder)
 - Input: latent variables
 - Output: parameters voor datadistributie $p_{\varphi}(x|z)$
- Parameters θ en φ zijn weights en biases van de neural networks
 - Optimaliseer weights + biases om de ELBO te maximaliseren

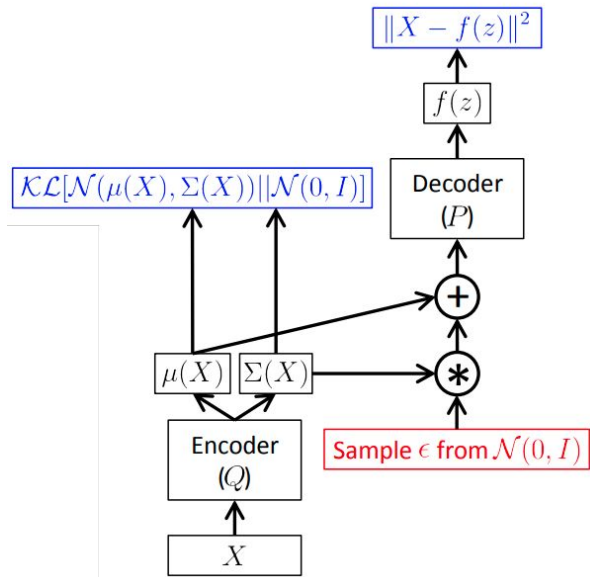


Perspectieven verbinden

- Approximate posterior $q_{\theta}(z|x, \lambda)$ wordt geparameteriseerd met een inference network (encoder)
 - Input: data x
 - Output: parameters λ
- Likelihood $p(x|z)$ wordt geparameteriseerd met een generative network (decoder)
 - Input: latent variables
 - Output: parameters voor datadistributie $p_{\varphi}(x|z)$
- Parameters θ en φ zijn weights en biases van de neural networks
 - Optimaliseer weights + biases om de ELBO te maximaliseren

$$\begin{aligned}\text{ELBO}(\theta, \varphi) &= \sum_{i=1}^N \mathbb{E}_q[\log p(x_i|z)] - \text{KL}(q_{\lambda}(z|x_i) \parallel p(z)) \\ \text{ELBO}(\theta, \varphi) &= -\mathcal{L}(\theta, \varphi)\end{aligned}$$

Model



Reconstructies

00 11 22 33 44

55 66 77 88 99

00 11 22 33 44

55 66 77 88 99

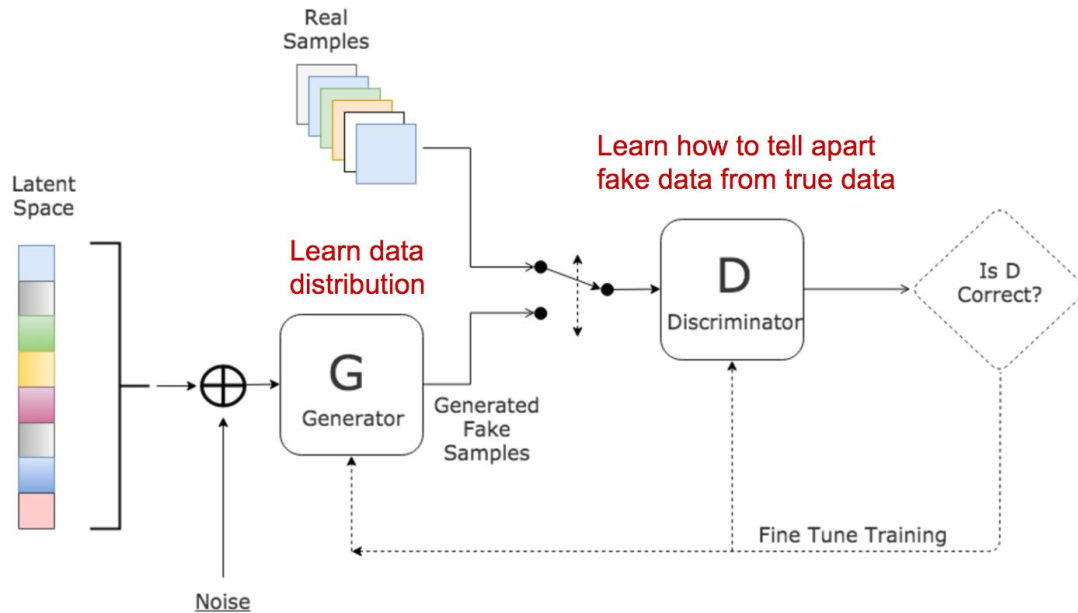
Generative Adversarial Networks



Generative Adversarial Network

- Leert de distributie van een dataset door twee neural networks tegen elkaar te laten strijden
 1. Generator: Genereert afbeeldingen die lijken op de dataset
 2. Discriminator: Probeert te ontdekken of de gegenereerde afbeeldingen echt of nep zijn
- Ze blijven elkaar proberen te verslaan
- Beide netwerken worden steeds beter
- Na een tijd maakt de generator afbeeldingen die niet te onderscheiden zijn van de echte dataset

Model





Objectives

1. Discriminator $D(x)$: Voorspelt de kans dat de input x van de echte dataset is gekomen
2. Generator $G(z)$: Verandert noise z naar een afbeelding



Objectives

1. Discriminator $D(x)$: Voorspelt de kans dat de input x van de echte dataset is gekomen
 - Maximaliseert de kans dat het juiste label wordt voorspeld voor zowel training examples als afbeeldingen die door de generator zijn gegenereerd
2. Generator $G(z)$: Verandert noise z naar een afbeelding
 - Minimaliseert de kans dat de discriminator kan voorspellen dat wat hij genereert nep is



Objectives

1. Discriminator $D(x)$: Voorspelt de kans dat de input x van de echte dataset is gekomen
 - Maximaliseert de kans dat het juiste label wordt voorspeld voor zowel training examples als afbeeldingen die door de generator zijn gegenereerd
2. Generator $G(z)$: Verandert noise z naar een afbeelding
 - Minimaliseert de kans dat de discriminator kan voorspellen dat wat hij genereert nep is

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

Objectives

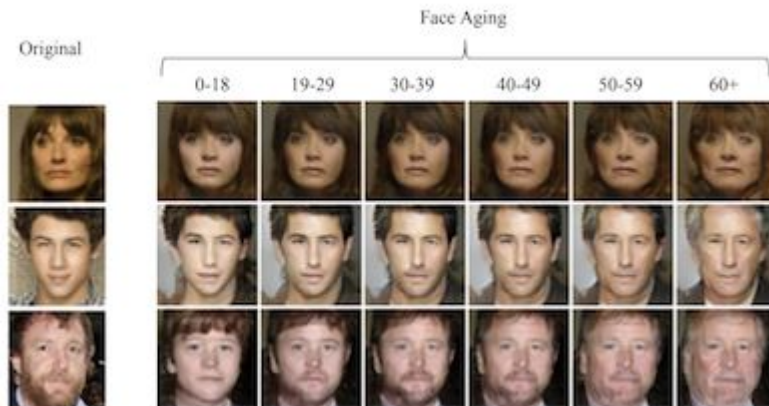
1. Discriminator $D(x)$: Voorspelt de kans dat de input x van de echte dataset is gekomen
 - Maximaliseert de kans dat het juiste label wordt voorspeld voor zowel training examples als afbeeldingen die door de generator zijn gegenereerd
2. Generator $G(z)$: Verandert noise z naar een afbeelding
 - Minimaliseert de kans dat de discriminator kan voorspellen dat wat hij genereert nep is

Voorspellingen van de
discriminator op de echte dataset
moeten zo dicht mogelijk bij 1
zitten

Voorspellingen van de
discriminator op de generator data
moeten zo dicht mogelijk bij 0
zitten

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)} [\log D(x)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

Outputs



Conclusie & Applicaties



Conclusie

- Generative models worden gebruikt om nieuwe samples te maken die van dezelfde distributie als de training data komen
 - Image super resolution
 - Kunst
- VAEs gebruiken een encoder-decoder structuur om samples te genereren
 - Encoder:
 - NN perspectief: Neural network dat representatie z van data x geeft
 - PM perspectief: Inference network dat de parameters van de variational posterior van latent variables z geeft $\rightarrow q(z|x)$
 - Decoder:
 - NN perspectief: Neural network dat de data x leert te reconstrueren gegeven een representatie z
 - PM perspectief: Generative network dat de parameters van de likelihood distribution geeft $\rightarrow p(x|z)$
 - Samples hebben vaak een lage kwaliteit
- GANs
 - Twee netwerken leren elkaar te verslaan totdat er samples van hoge kwaliteit gegenereerd kunnen worden

Applicaties

- Voorbeelden voor image datasets genereren (slaapkamers)
- Image-to-image translation



Deepfakes



- Bestaande afbeeldingen en video's worden gecombineerd
- Fake news
- Deep fake detection

Assignment 2

<https://git.io/JvB5u>
