Procedural Content Generation

Abe Vos

Juni 2022

Vorige keer

Procedurele Content Generatie

- Parameters
- Stochasticiteit
- Procedurele functie

Stochastische processen

- Modelleer proces met kansverdeling
- $p(X = x) \approx \hat{p}(X = x)$
- Neem steekproeven voor generatie

Simultane kansverdelingen

- Stochast met meerdere dimensies
  - $X = X_1, \cdots, X_D$
- Simultane kansverdeling over alle componenten
  - $p(X = x) = p(X_1 = x_1, \cdots, X_D = x_D)$
- Moeilijk te berekenen

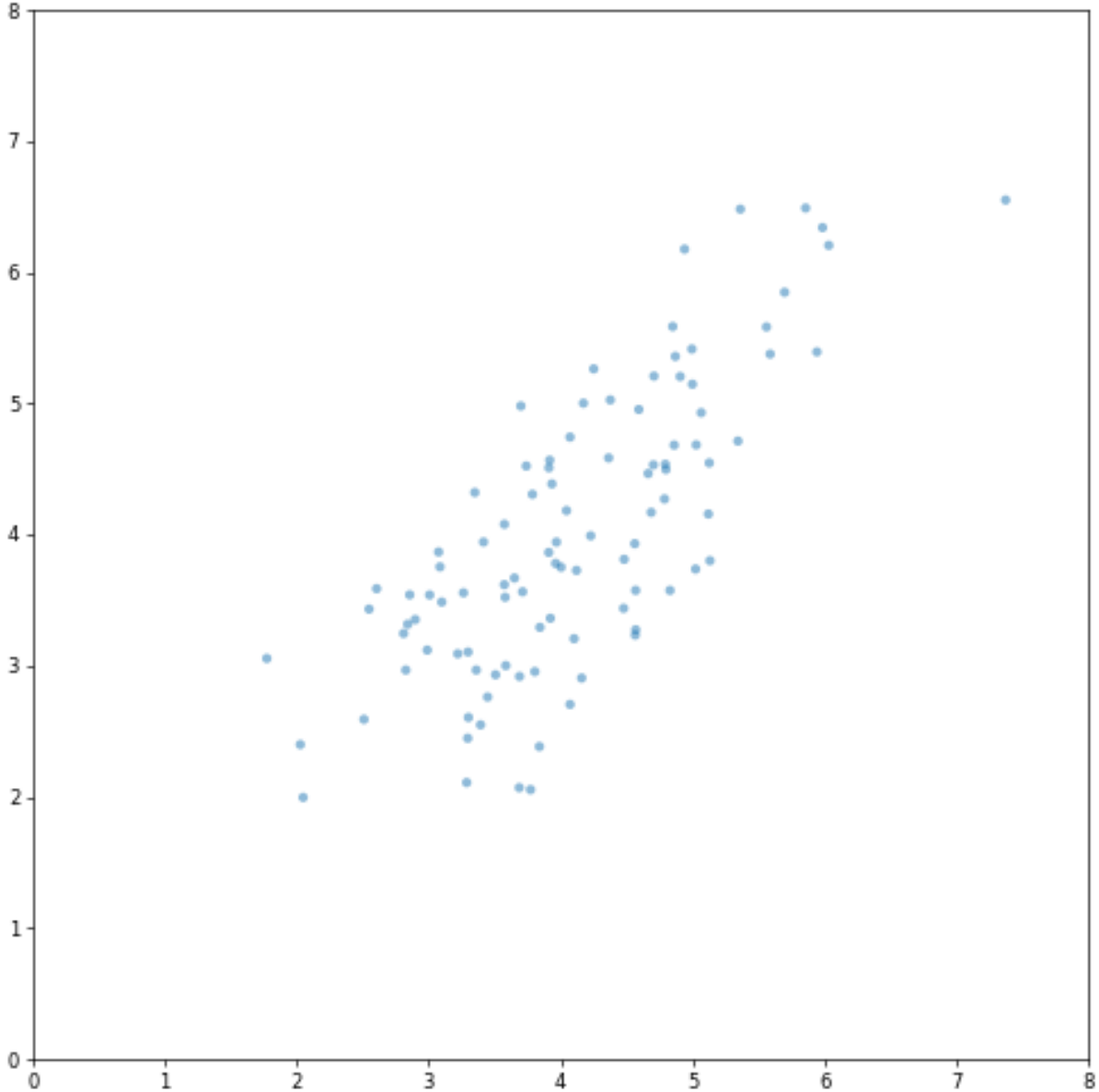
Markov Ketens

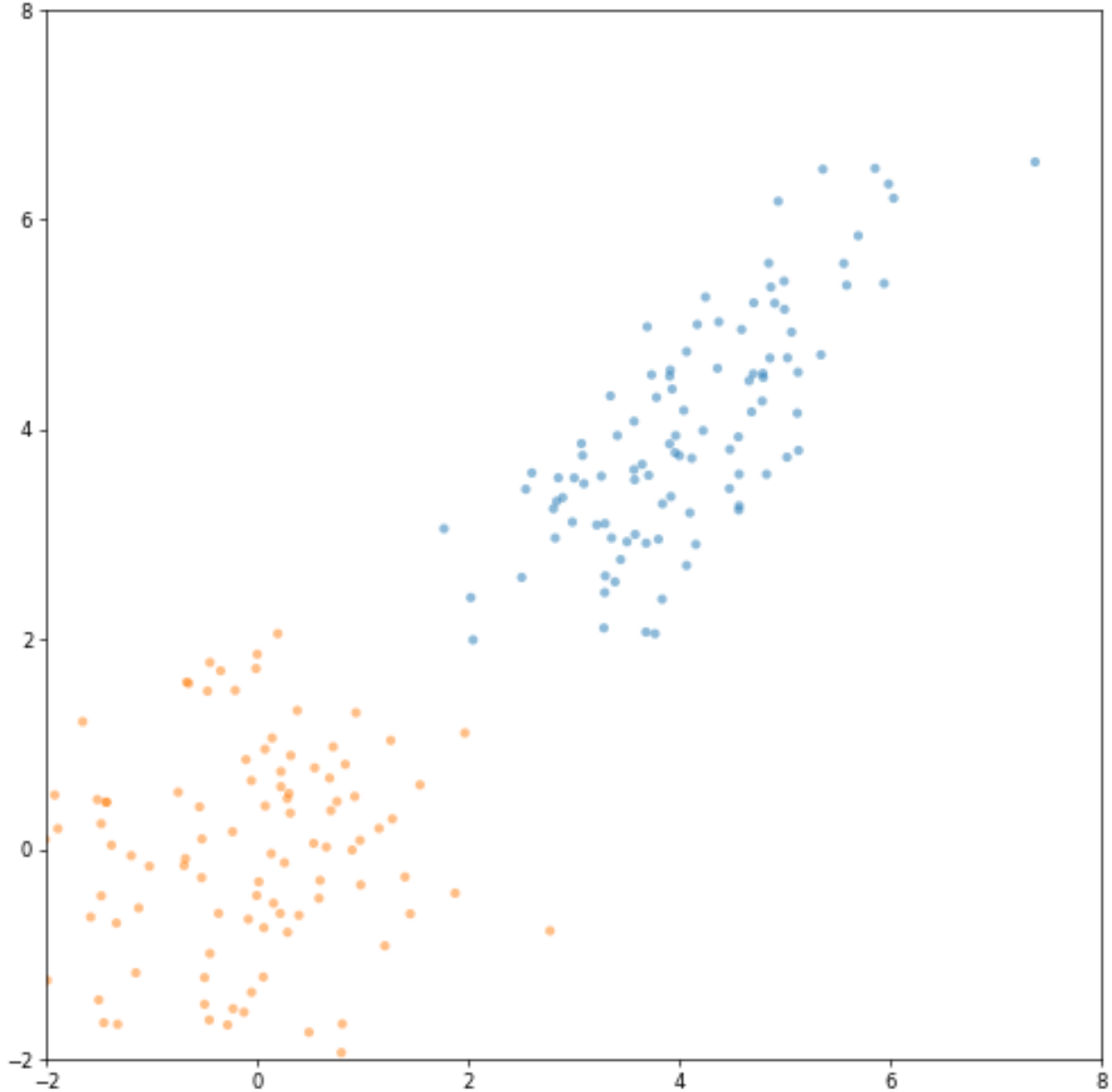
- Vervang simultane kansverdeling door voorwaardelijke kansverdeling
  - $p(X_1, X_2, \cdots, X_D) = p(X_1)p(X_2|X_1)\cdots p(X_D|X_{D-1})$
- Voorwaardelijke kansberekeningen
  - Bigrams/Ngrams in tekst
  - Makkelijk te berekenen
- Nuttig voor reeksen

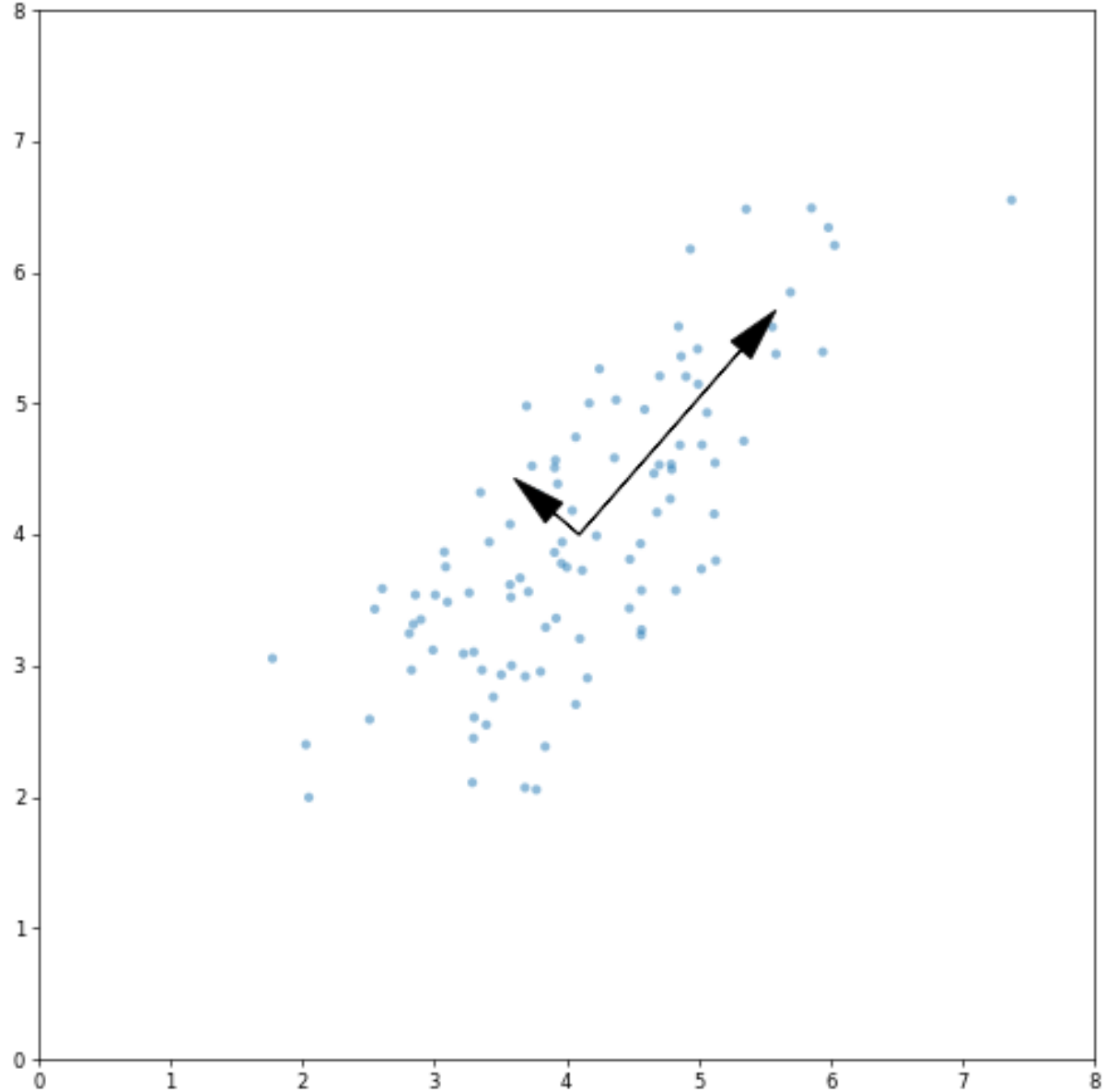
Volgende stap

- Simultane kansverdeling
- Reeks data: voorwaardelijke kansverdeling
- Afbeeldingen: transformeer simpele kansverdeling

Principal Component Analysis

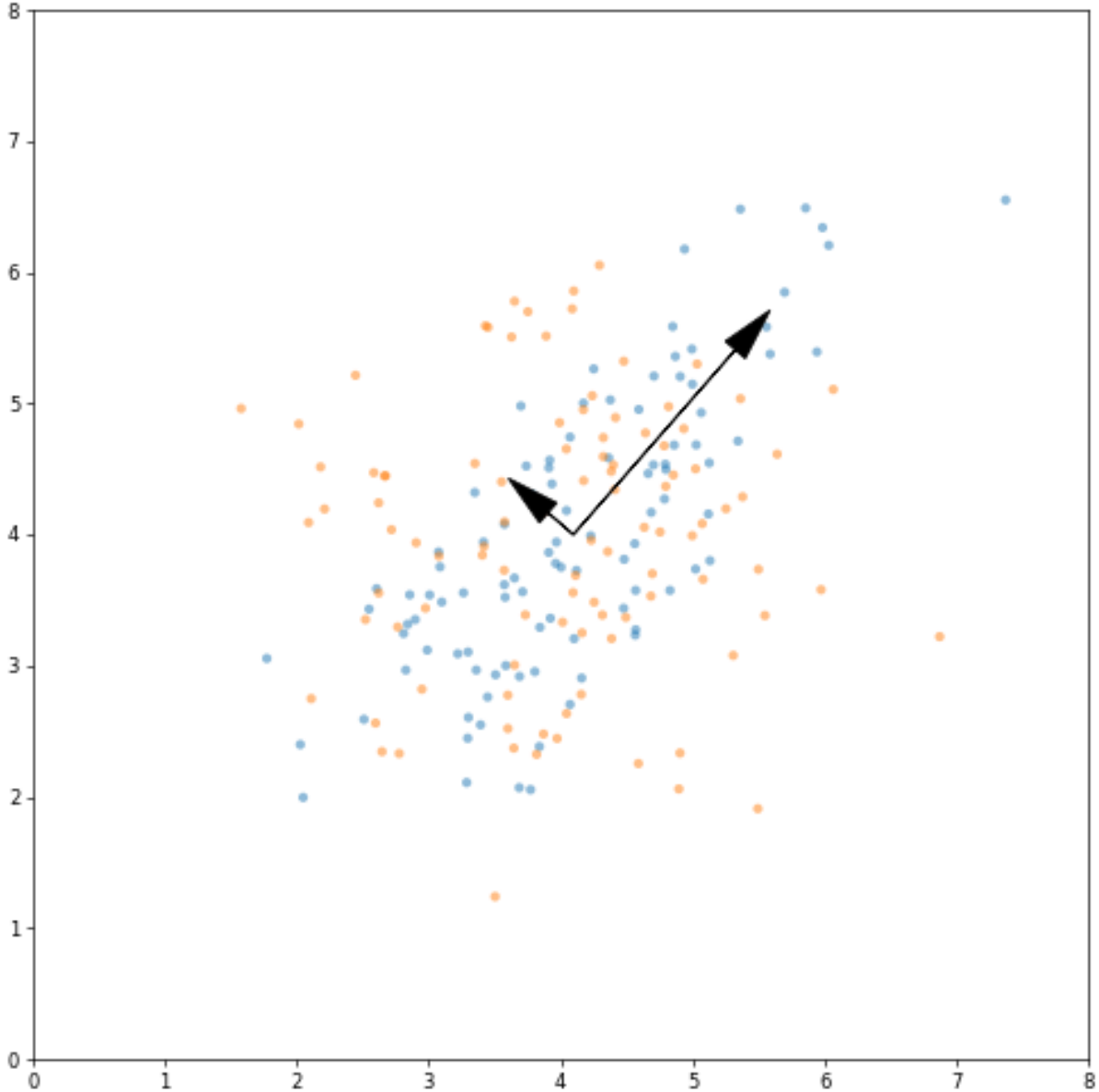

0
1
2
3
4
5
6
7
8
0
1
2
3
4
5
6
7
8

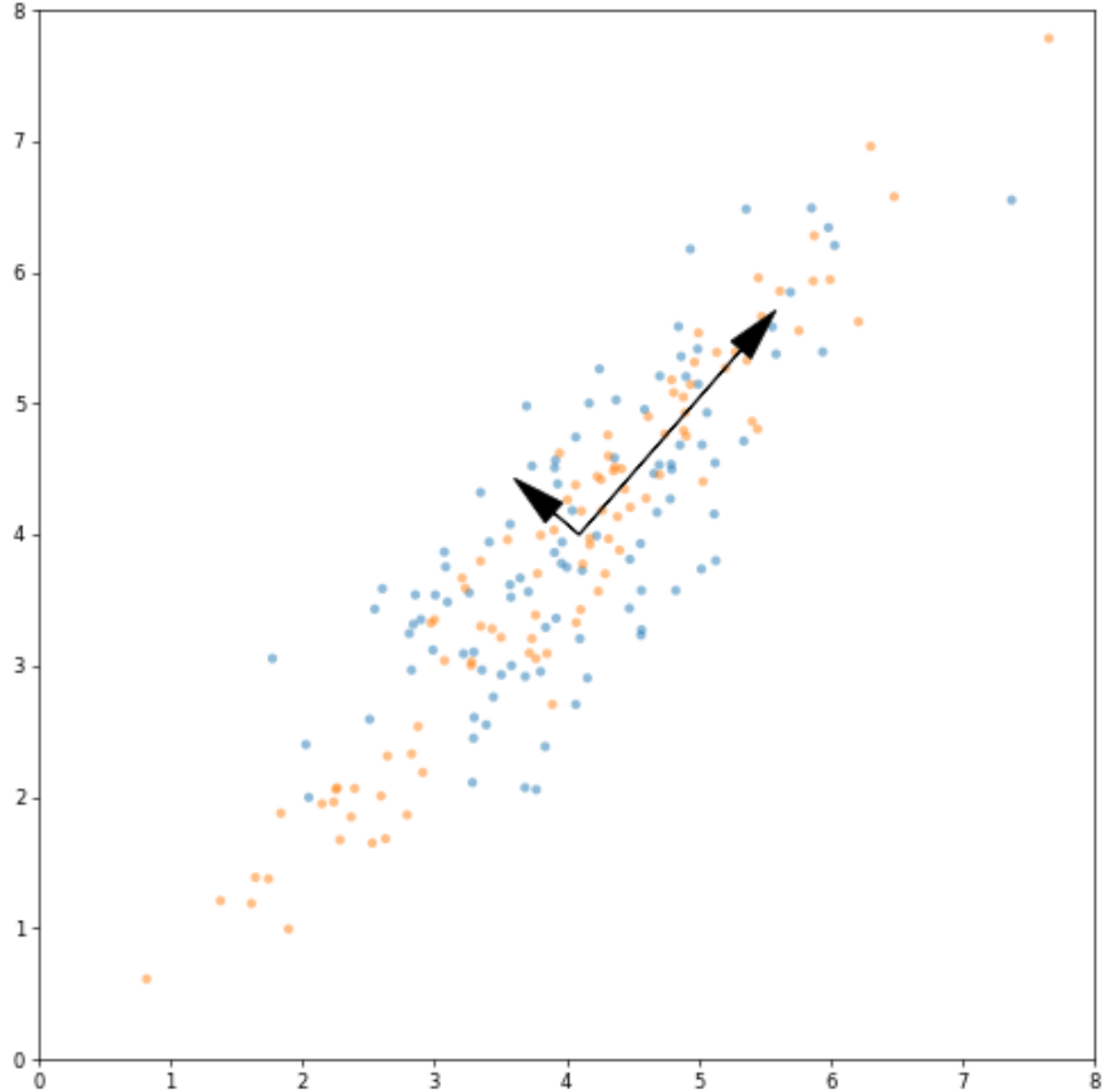

8
6
4
2
0
−2
−2
0
2
4
6
8


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

Principiële componenten

- Projecteer datapunten op vector
- Vind vector die variantie van datapunten maximaliseert
- Orthogonale vectoren


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


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

Getransformeerde datapunten

- Principiële componenten $\mathbb{v}$ en $\mathbb{w}$
- Genereer normaal verdeeld punt $x = (x_1, x_2)$
- Transformeer punt: $x_1\mathbb{v} + x_2\mathbb{w}$

Latente vector model

- Generatief model
- Neem steekproef van eenvoudige kansverdeling
- Transformeer punt naar steekproef van ingewikkelde kansverdeling
- Transformatie kan in beide richtingen

Eigenfaces

- PCA voor gezichten
- Afbeelding als vector

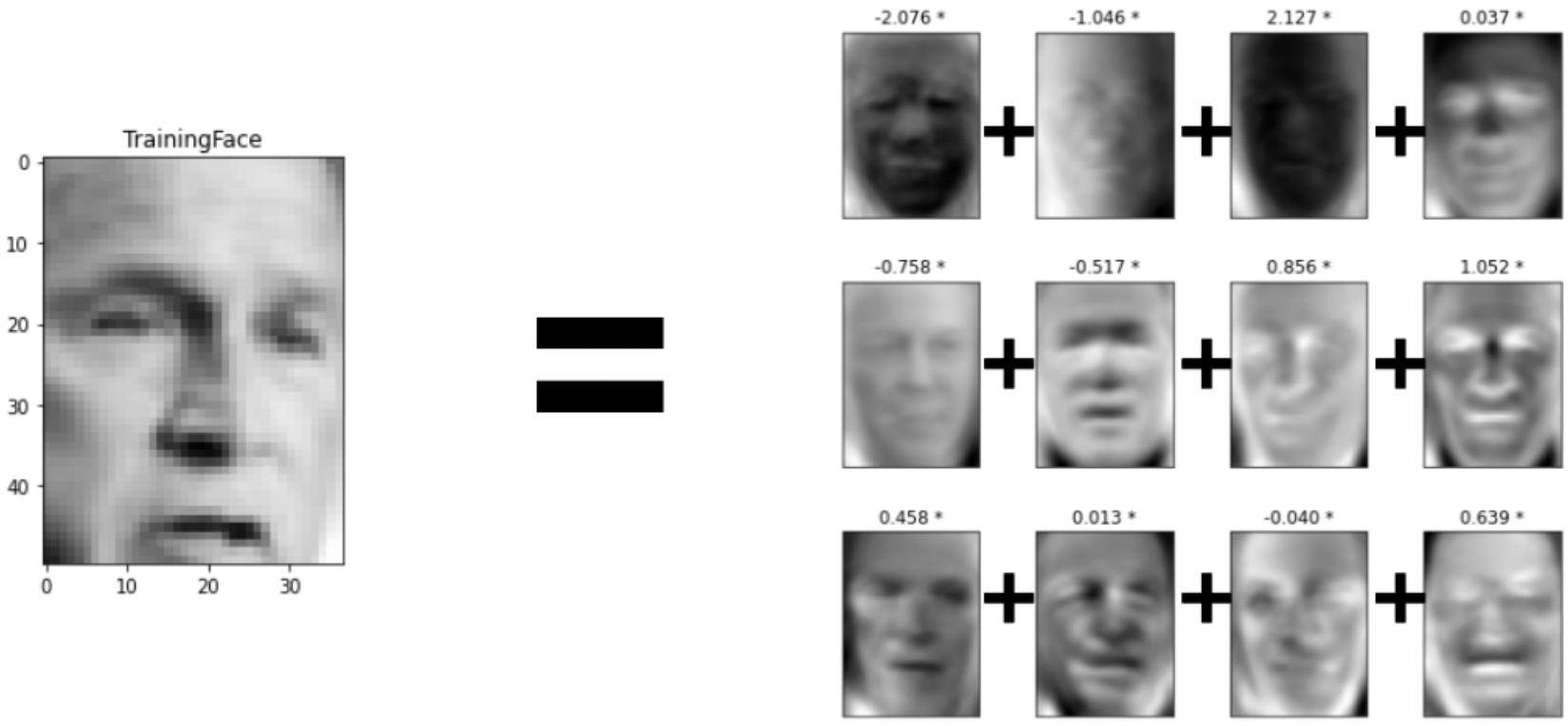


Figure 6: Decompositie van gezicht

Autoencoders

PCA

- Lineair
- Makkelijk te berekenen
- Beperkte resultaten
- Schaalt niet goed naar grote datasets

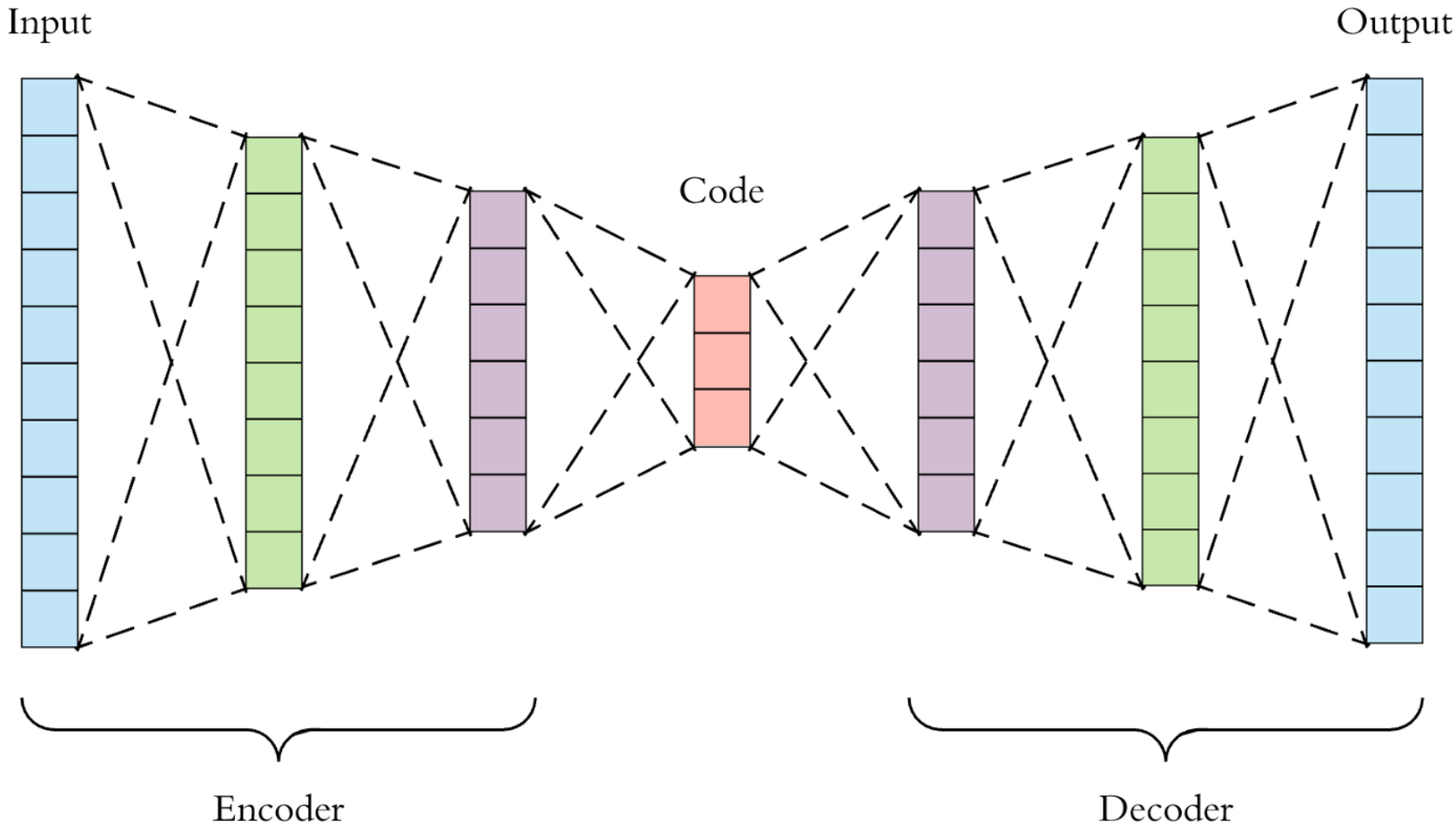


Figure 7: Autoencoder architectuur

Autoencoders

- Encoder: input naar latente vector
- Decoder: latente vector naar output
- Train model om output op input te laten lijken

Informatie bottleneck

- Dwing autoencoder om alleen belangrijke informatie te behouden
- Latente vector is “samenvatting” van input

Voorbeelden van bottlenecks

- Dimensies latente vector $\ll$ dimensies input/output
- Voeg ruis toe aan verborgen lagen
  - Diffusion model
- Punten in latente vector volgen kansverdeling
  - Bijvoorbeeld normaal verdeling

Variational Autoencoder

- Encoder produceert twee outputs
  - Parameters voor normaalverdeling
- Error functie heeft twee componenten
  - Verschil tussen input en output $((x - f(x))^2)$
  - Kullback-Leibler divergentie van parameters uit encoder

Kullback-Leibler Divergentie

- Genereer punten met kansverdeling $q(X = x)$
- Doe alsof punten uit kansverdeling $p(X = x)$ komen
- $D_{\mathrm{KL}}\left(p(X) \parallel q(X)\right)$
  - Hoe verbaast zijn we over de resultaten van $q(x)$ als we $p(x)$ verwachten?

Voordelen van bottleneck

- Mogelijke afbeeldingen $\gg$ mogelijke latente vectoren
- Prop afbeeldingen uit dataset efficiënt in latente ruimte
- Meeste punten in latente ruimte komen overeen met datapunten

Generatie met autoencoder

- Genereer punt in latente ruimte
  - Vaak met standaard normaalverdeling
- Transformeer punt met decoder model

Unsupervised learning

- Afbeeldingen zonder labels
- Latente ruimte omschrijft abstracte eigenschappen van input
  - Encoder leert nuttige informatie uit input te halen
- Gebruik encoder als basis voor neuraal netwerk
  - Transfer learning

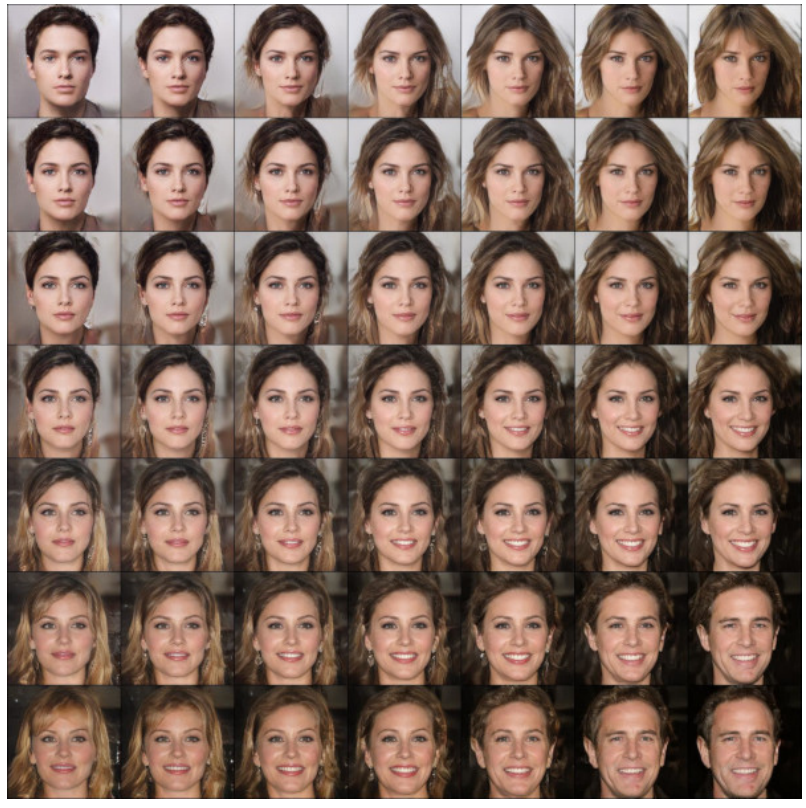
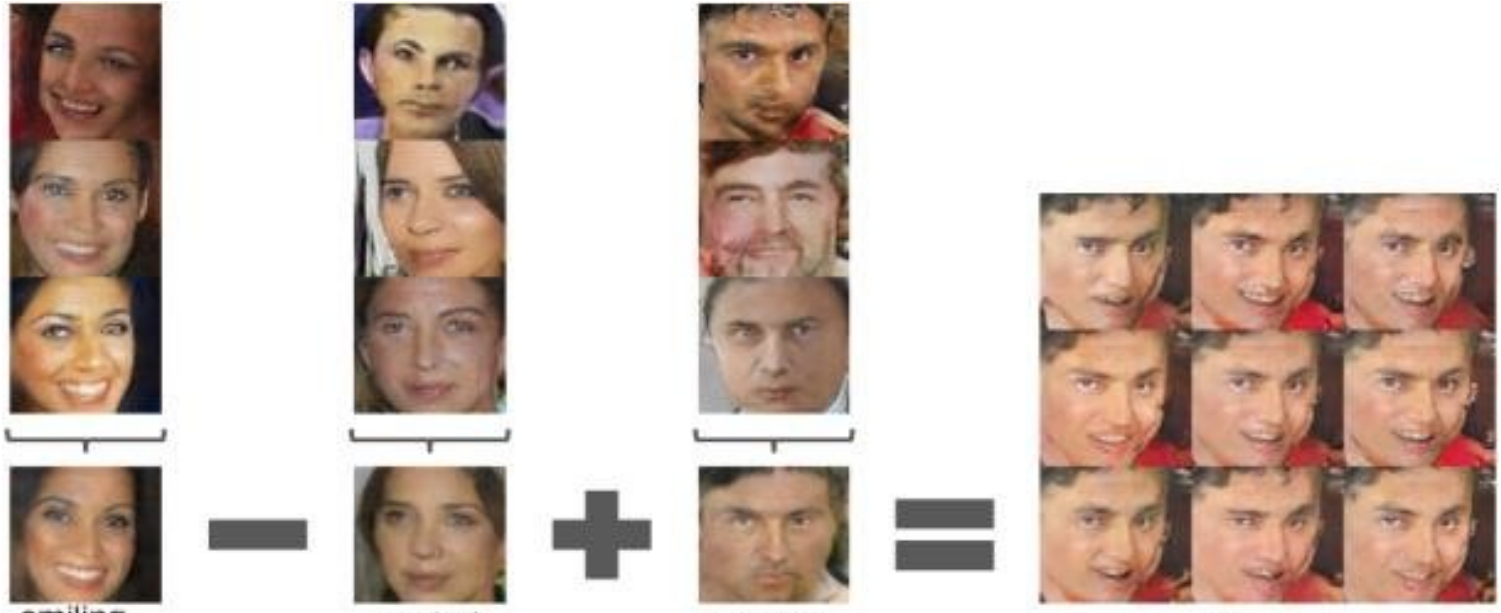
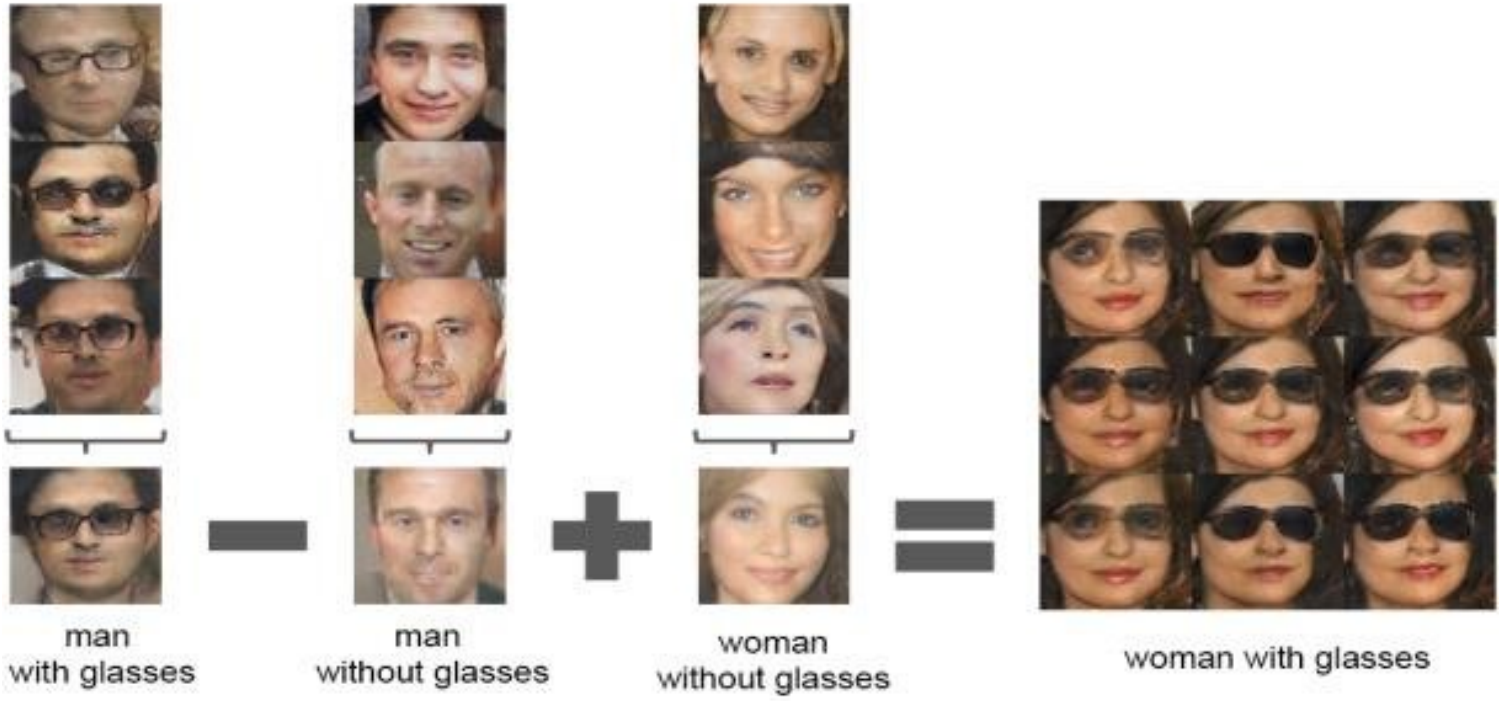


Figure 8: Interpolatie in latente ruimte


smiling woman
neutral woman
neutral man
smiling man


man with glasses
man without glasses
woman without glasses
woman with glasses

Convolutional Neural Networks

Neurale netwerken voor afbeeldingen

- Input: afbeelding 32x32 pixels; 3 kleuren
  - 3072 input neuronen
- Eerste verborgen laag: 1000 neuronen
- Ruim drie miljoen parameters!

Lokale features

- Afbeeldingen zijn opgebouwd uit simpelere elementen
- Afbeelding van typisch gezicht
  - Twee ogen
  - Neus
  - Mond
- Simpel neuraal netwerk moet dezelfde “features” leren herkennen op verschillende plekken

Convolutional Layer

- Herbruik dezelfde parameters op verschillende plekken in de afbeelding
- Minder parameters
- Features zijn onafhankelijk van positie

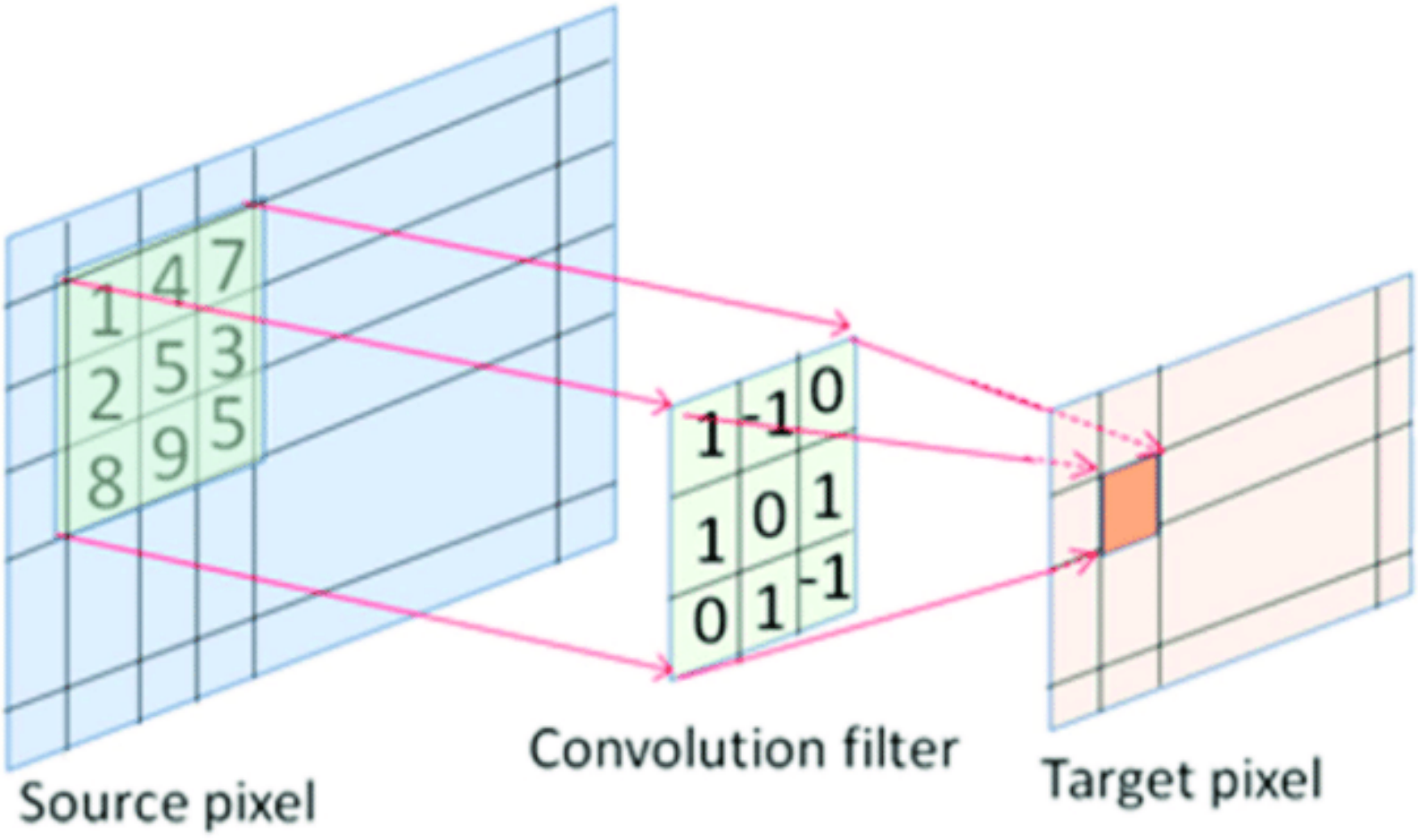


Figure 10: Convolutional Layer

Convolutional Layer

- Schuif convolution filter/kernel over de hele afbeelding
- Bereken dot-product tussen filter en onderliggende pixels op elke positie
- Sla resultaten van dot-producten op in feature map
- Herhaal voor alle features/verborgen neuronen

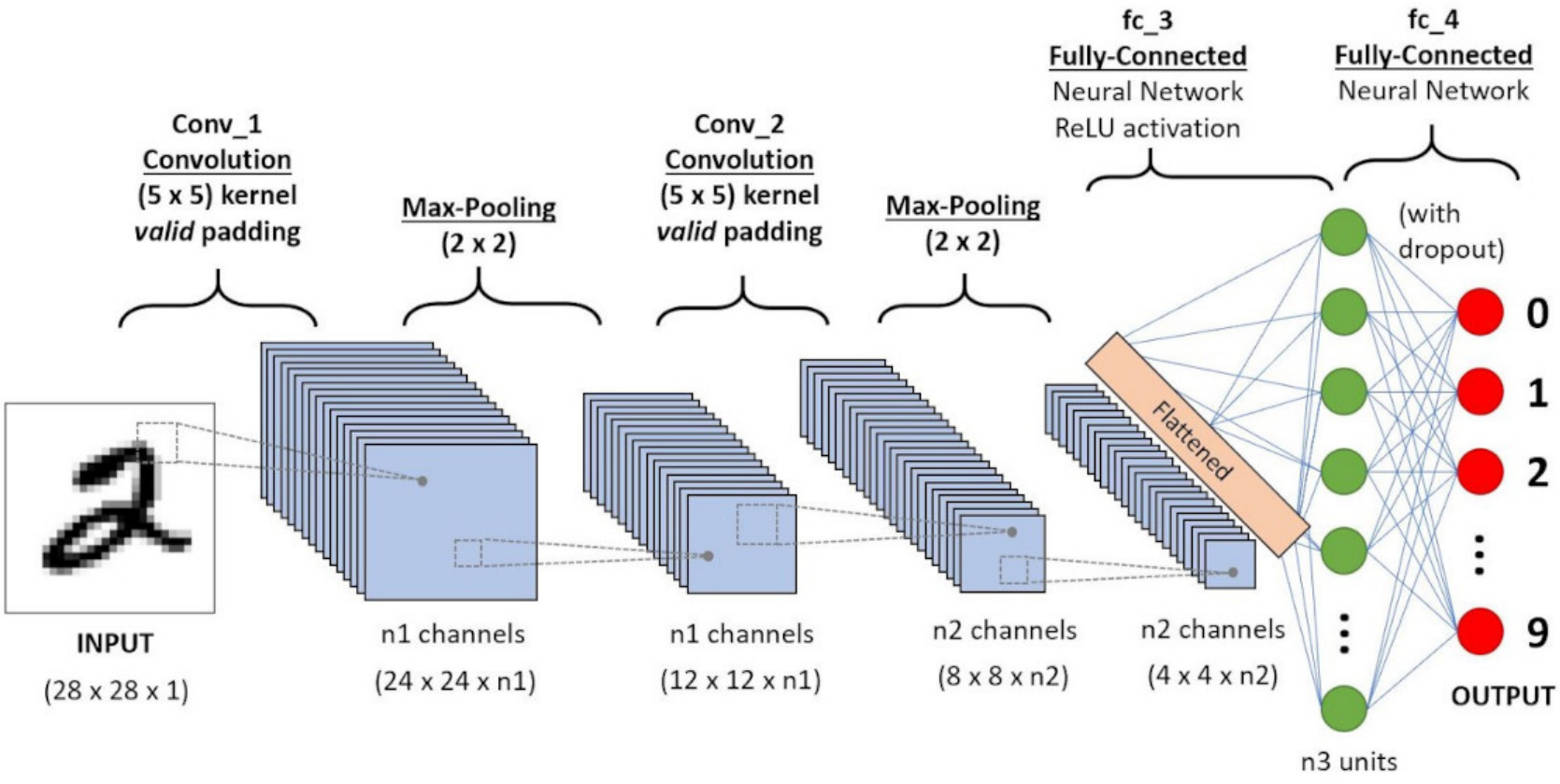


Figure 11: Convolutional Neural Network

Wat leert een CNN?

- Algemene features om een afbeelding uit op te bouwen
- Hiërarchische structuur
  - Onderste lagen: basale vormen
  - Middelste lagen: complexe vormen, textuur
  - Bovenste lagen: objecten

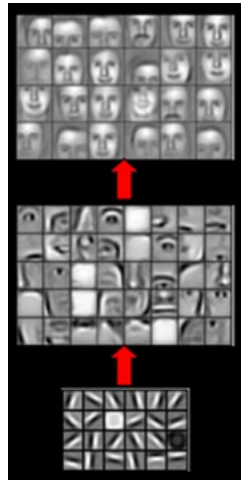


Figure 12: Geleerde features van CNN

Deconvolution

- Omgekeerde operatie van convolution
- Voorspel ruimtelijke “low-level” features gegeven “high-level” features
- Gebruikt om afbeeldingen te genereren
  - Decoder van autoencoder

Generative Adversarial Networks

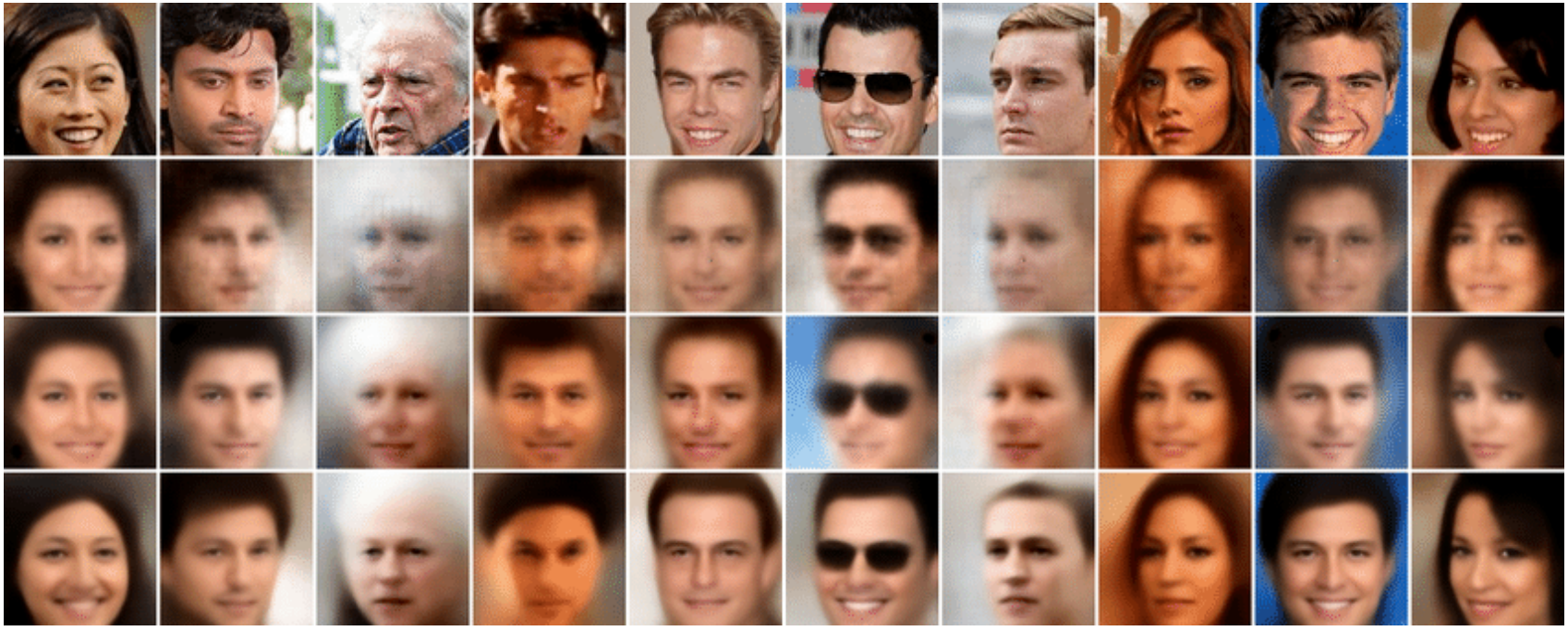


Figure 13: Wazige reconstructies van de VAE

Generative Adversarial Network

- Generator produceert afbeeldingen
- Discriminator voorspelt of een afbeelding gegenereerd is
- Zero-sum game
  - Beide netwerken presteren beter als het andere netwerk slechter presteert

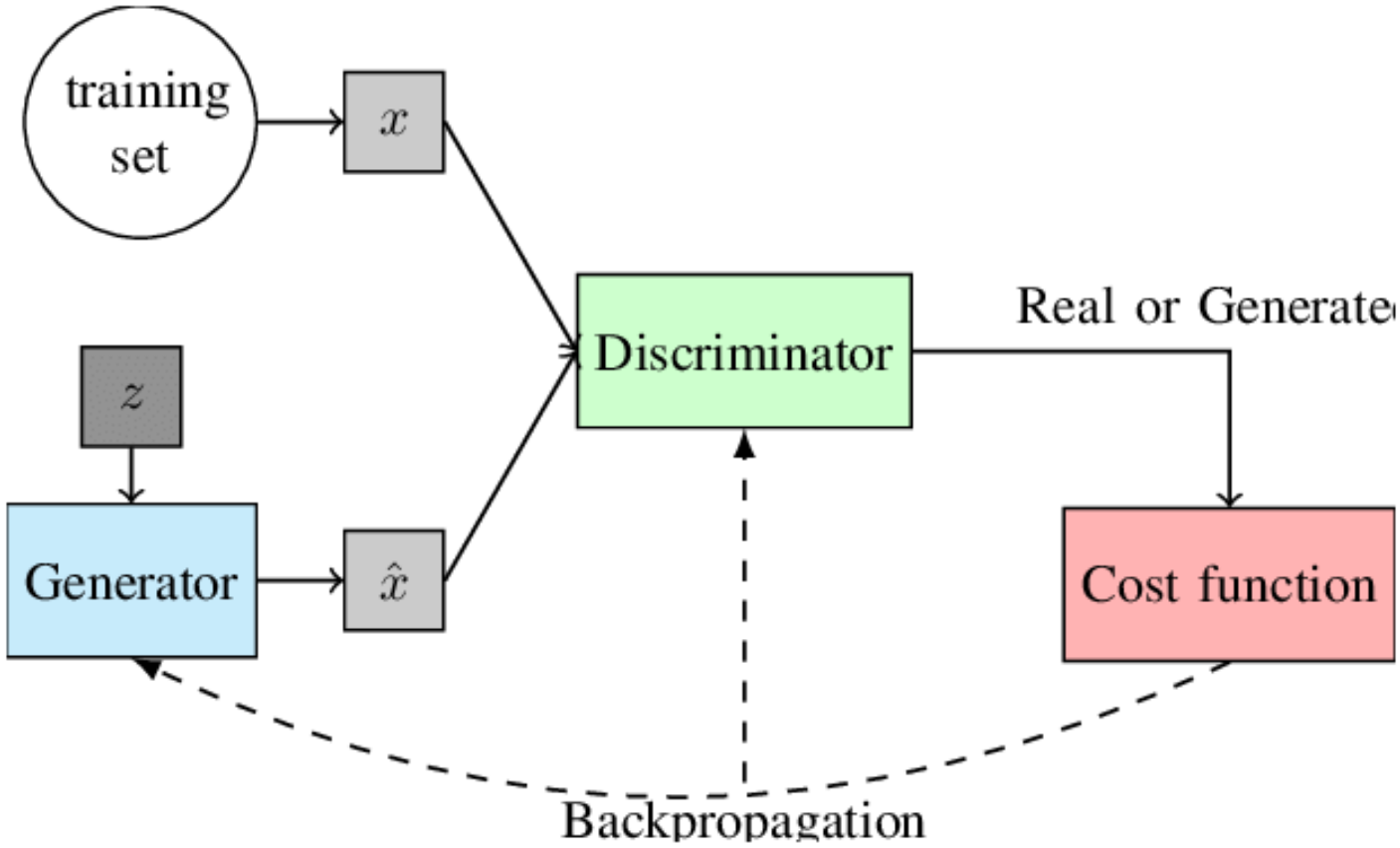


Figure 14: Architectuur van de GAN

Generator

- Neuraal netwerk met deconvolution lagen
- Input: vector z met waardes gegenereerd uit een standaard normaal verdeling
- Decoder

Discriminator

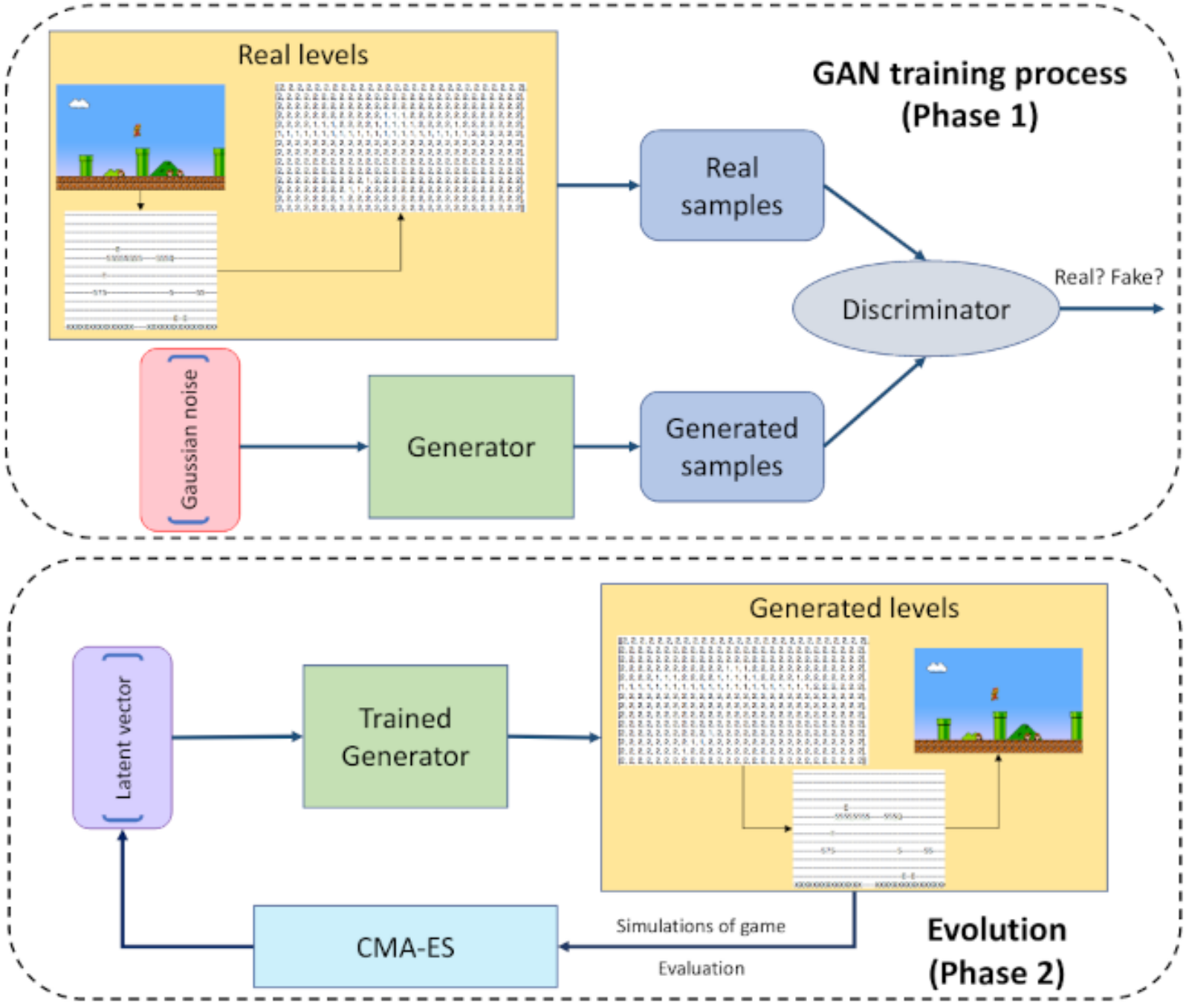
- CNN met 1 output
  - 0 voor de voorspelling “gegenereerd”
  - 1 voor de voorspelling “echt”
- Input: data uit de dataset of gegenereerd door generator

Procedurele generatie

- Noise, parameters, procedurele functie
- GAN gebruikt noise en procedurele functie
- GAN heeft *geen* encoder
  - Geen controle over gegenereerd resultaat
  - Geen parameters

MarioGAN

- Genereer representatie van Mario level met GAN
- Gebruik evolutie strategie om speelbaar level te vinden
  - CMA-ES
- Denk terug aan wenselijke eigenschappen
  - Variatie (door random sampling)
  - Non-typische resultaten (door GAN)
  - Werkt met game mechanics (door CMA-ES)


Real levels
GAN training process
(Phase 1)
Real
samples
Discriminator
Real? Fake?
Gaussian noise
Generator
Generated
samples
Generated levels
Latent vector
Trained
Generator
CMA-ES
Simulations of game
Evaluation
Evolution
(Phase 2)

Conditional GAN

- Genereer afbeeldingen met noise *en* labels
- Genereer afbeeldingen van een bepaalde categorie
- Geeft controle over de output van generator

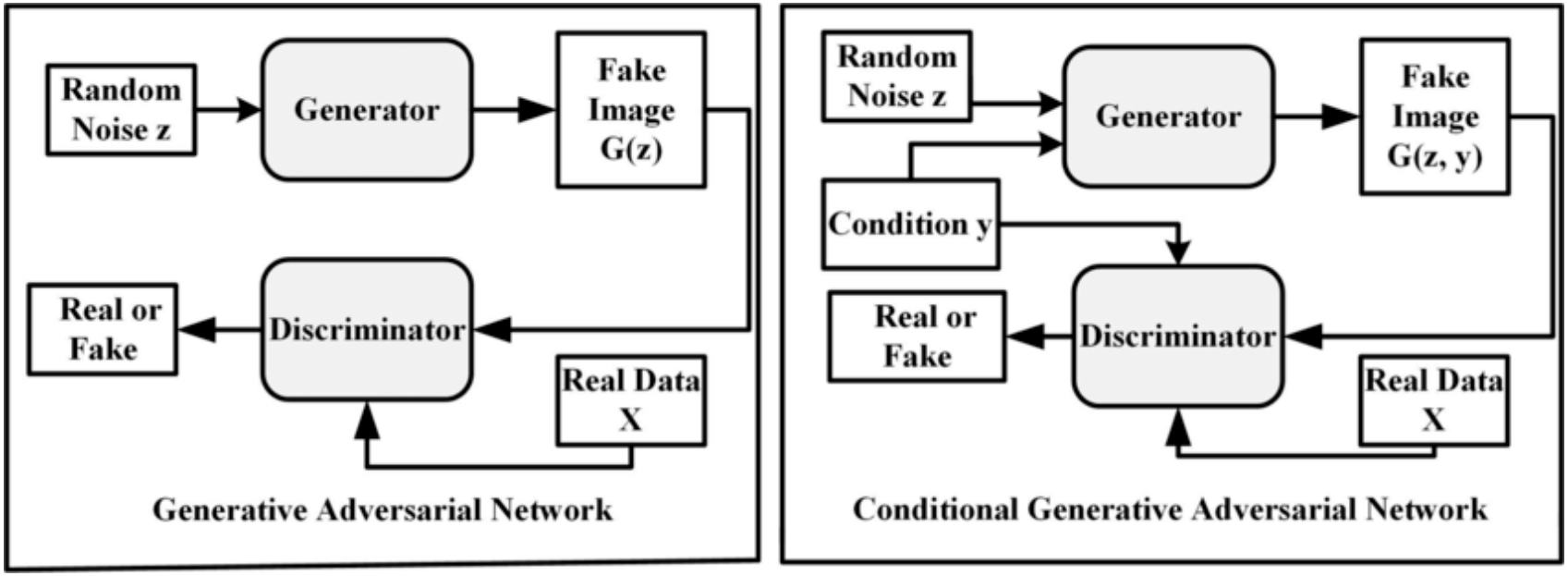


Figure 16: CGAN

Pix2pix

- “Labels” kunnen ook afbeeldingen zijn
- Genereer label afbeeldingen a.d.h.v. afbeelding in dataset
  - Bijvoorbeeld zwart-wit afbeelding van een kleurenafbeelding

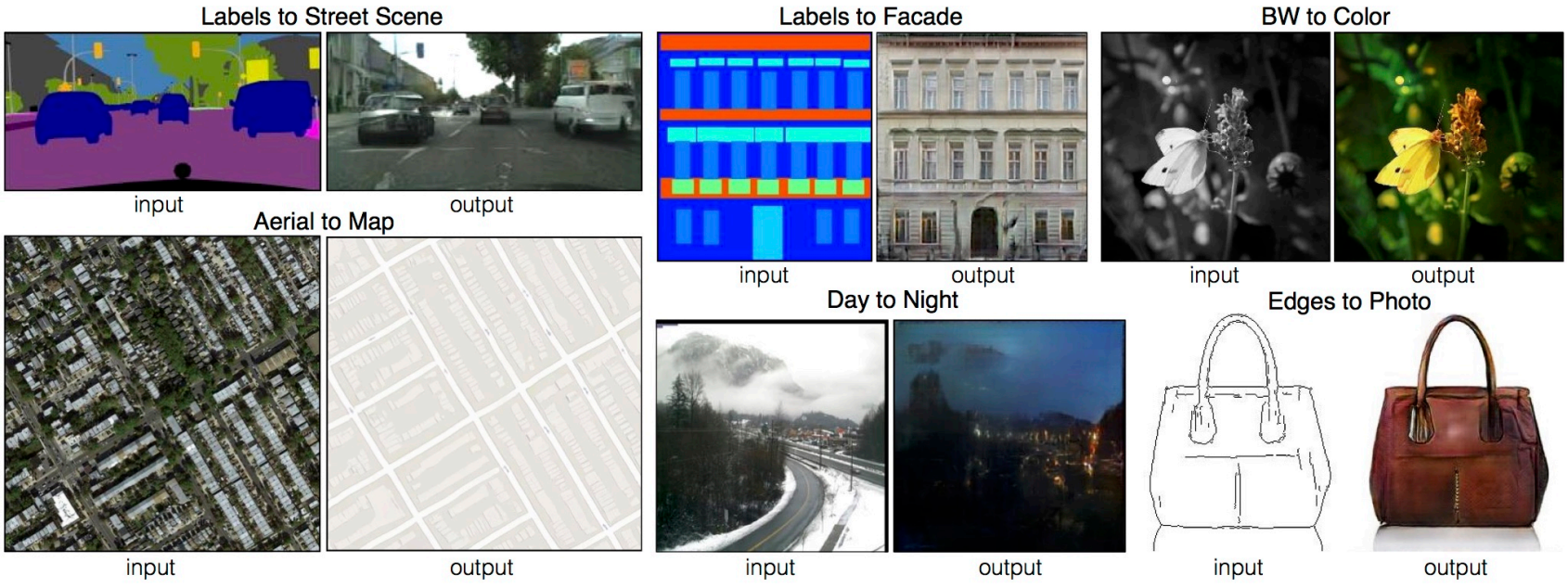


Figure 17: Toepassingen van pix2pix

Sketch2Map

- Genereer (hoogte)kaarten a.d.h.v. schetsen
- Training data: satelliet foto's en gegenereerde kaarten
- Gebruikt cGAN om schets in kaart te veranderen

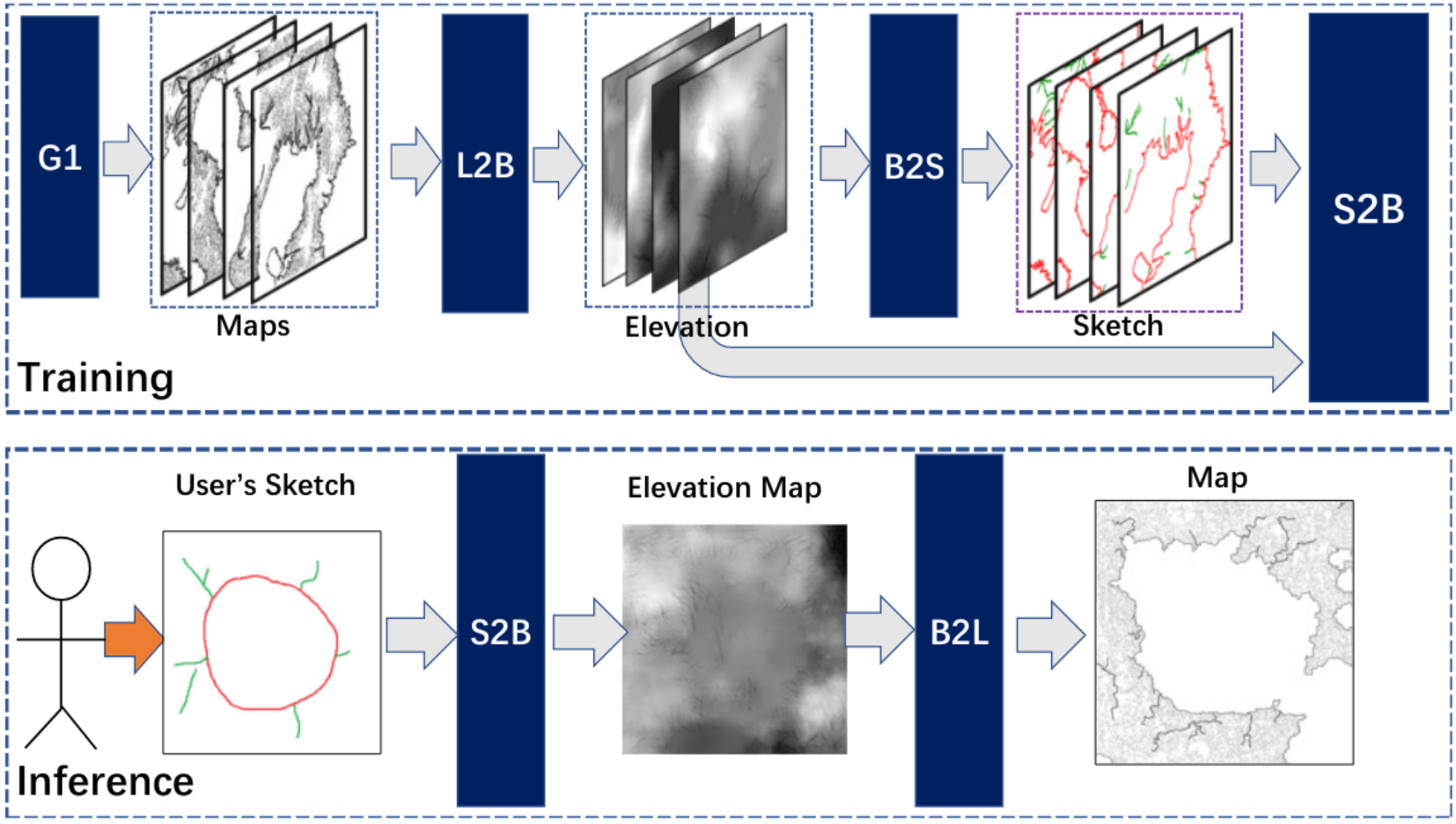


Figure 18: Data pipeline

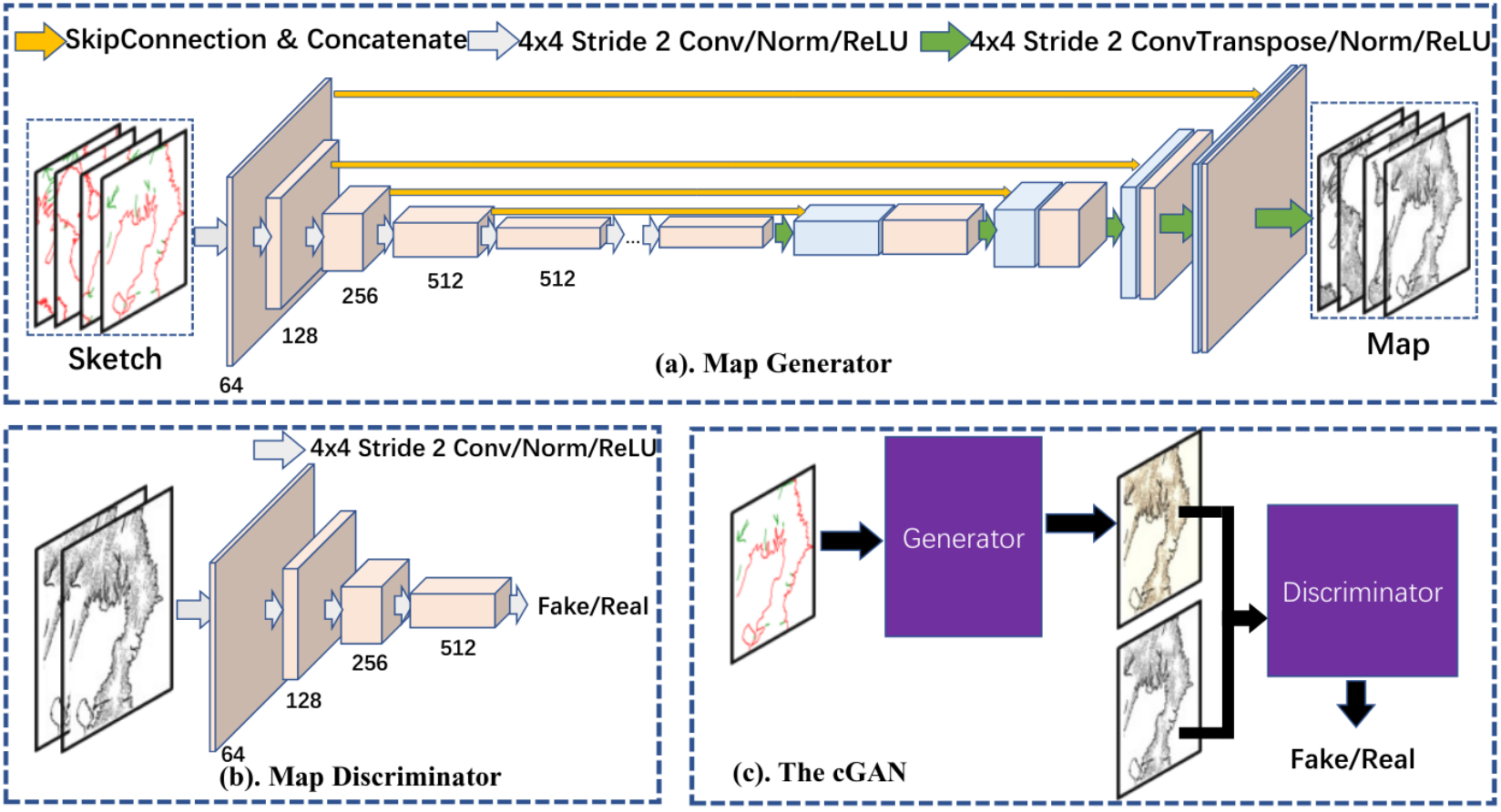
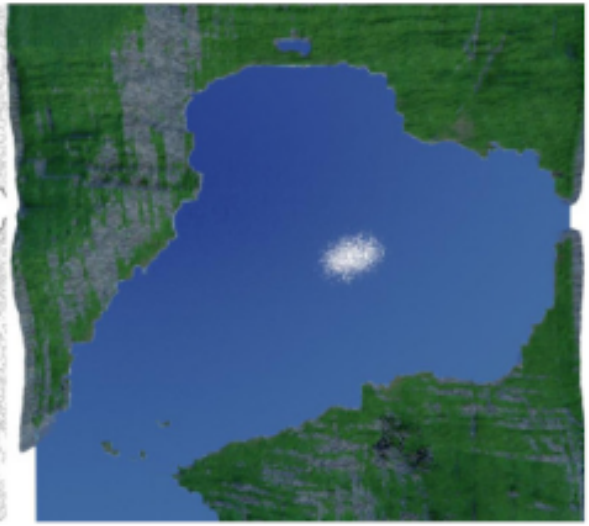


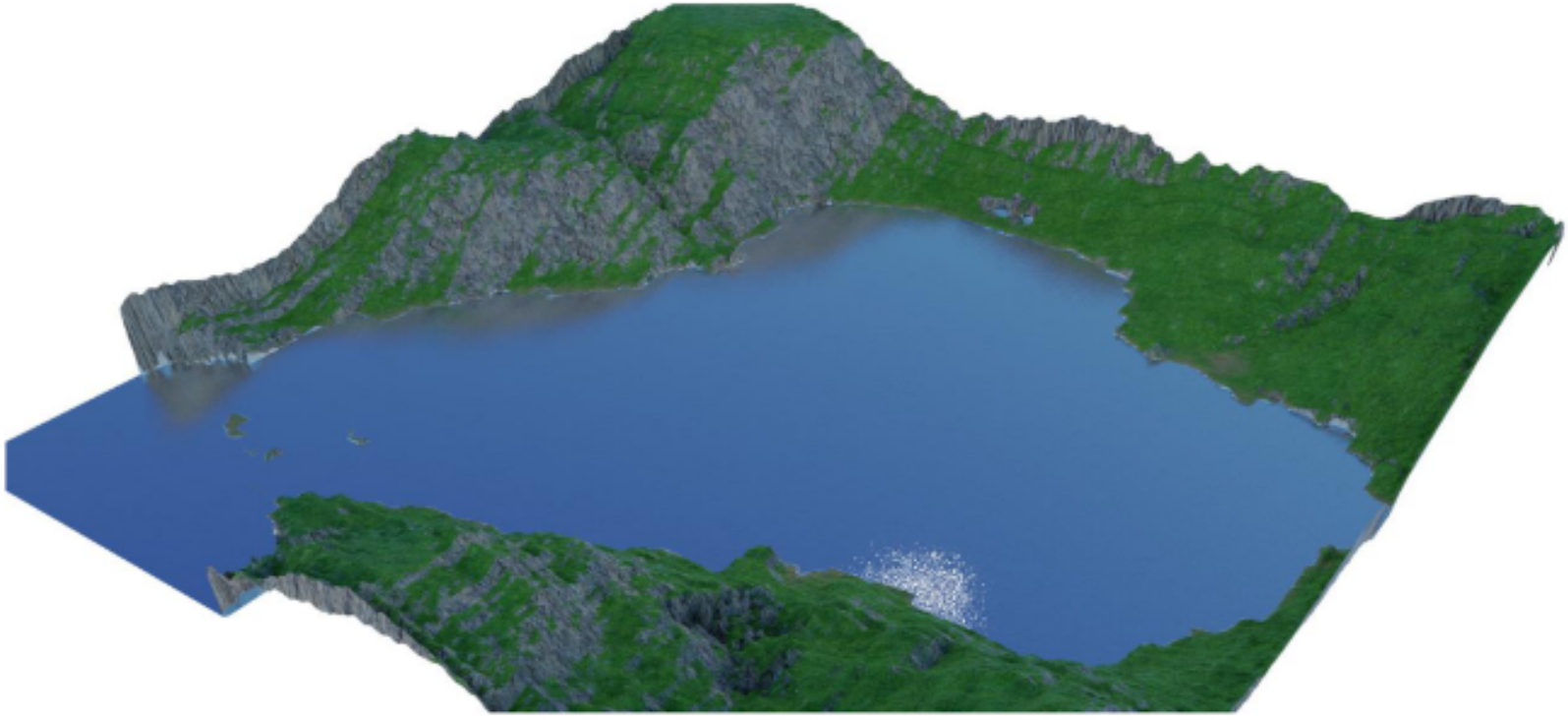
Figure 19: cGAN architectuur

(a). User generated sketch

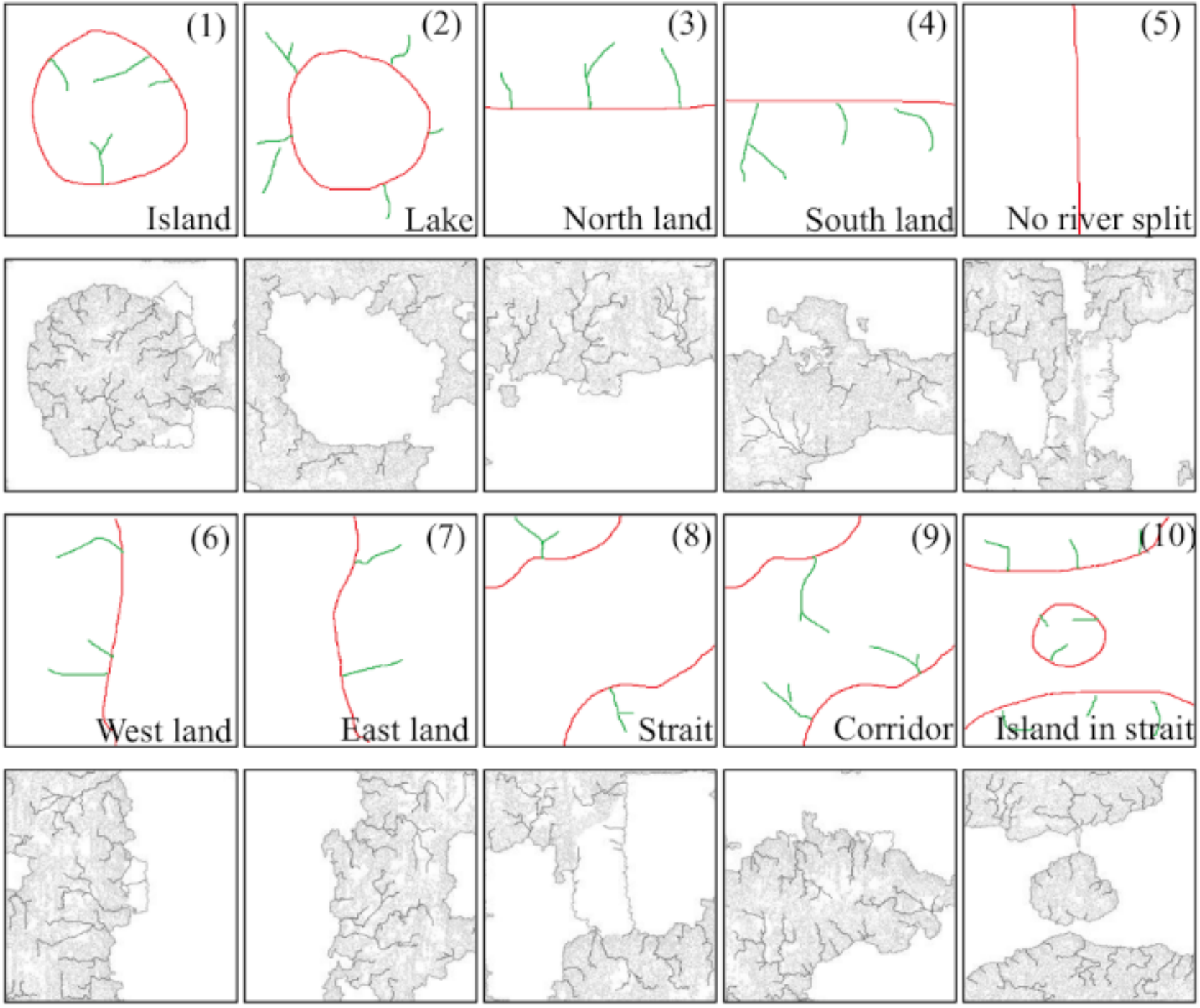
(b). 2D map generated with Sketch2Map



(c). 3D rendering result



(d). Perspective view of the generated map


(1)
Island
(2)
Lake
(3)
North land
(4)
South land
(5)
No river split
(6)
West land
(7)
East land
(8)
Strait
(9)
Corridor
(10)
Island in strait

Superresolutie

- Genereer hoge resolutie versie van input
- Training data: ongelabelde afbeeldingen met omlaag geschaalde versies
- Twee aanpakken:
  - Autoencoder
  - cGAN: (E)SRGAN

Figure 22: Max Payne

Figure 23: Originele Doom sprites

Figure 24: Superresolutie Doom sprites