Procedural Content Generation

Abe Vos

Mei 2022

Introductie

Wat is PCG?

- Algoritmische generatie van data
  - Beeld, geluid, tekst e.d.
- Imitatie van natuurlijk en/of artistiek proces

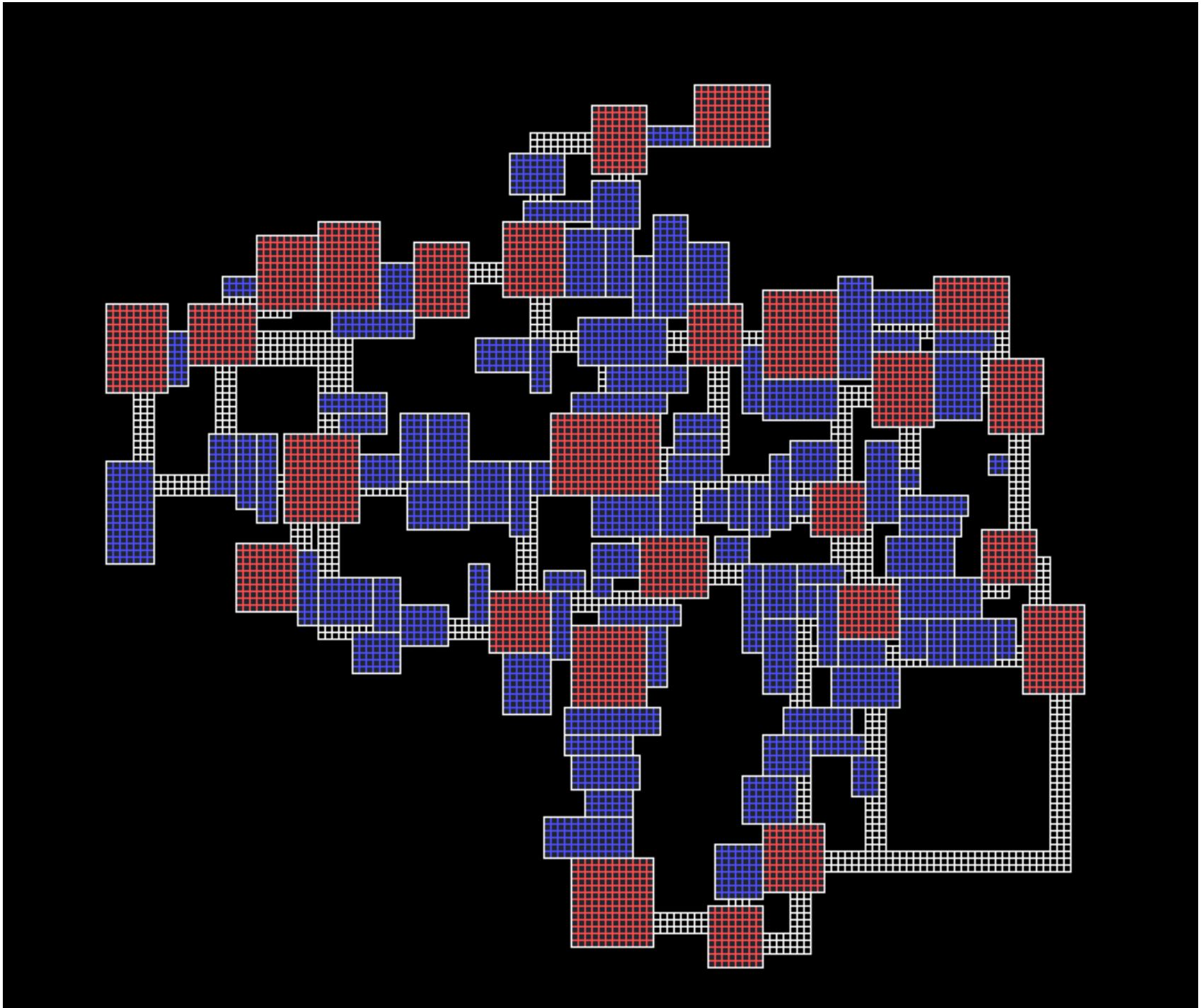
Voorbeeld: Map generation

- Genereer Perlin noise
- Bepaal cut-offs voor water/strand/bergen
- Schrijf kleuren/textures toe aan regio's

Figure 2: 3D map generation

Wenselijke eigenschappen

- Eindeloze variatie d.m.v. stochasticiteit
- Interessante/overtuigende resultaten
  - Resultaat moet er niet “gegenereerd” uitzien
- Sluit aan op game mechanics



Generatief model

- PCG heeft drie onderdelen:
  - Parameters
  - Onzekerheid
  - Generatieve functie
- Generatieve functie ontworpen a.d.h.v. regels
- Hetzelfde probleem als bij RL:
  - Vervang handgeschreven regels met kennis uit data

Machine Learning & PCG

Van de grond op

- Het resultaat X van een natuurlijk proces is stochastisch
  - Bijv.: boom/terrein/gezicht/wapen
- Proces is een kansverdeling over alle uitkomsten van X
  - $\hat{p}(X = x)$
- PCG is een vereenvoudigd model van het proces om uitkomsten te genereren
  - $p(X = x) \approx \hat{p}(X = x)$
  - Generatief model

Proces als kansverdeling

- Dobbelsteen is te omschrijven als natuurkundig proces
- Moeilijk te simuleren:
  - Gewicht van dobbelsteen
  - Wrijving tussen dobbelsteen en oppervlakte
  - Botsingen met omgeving
  - Bewegingen van de hand
- Modelleer uitkomsten als kansverdeling
  - Neem steekproeven van deze kansverdeling

Kansverdeling van dobbelsteen

x	$p(X = x)$
1	$\frac{1}{6}$
2	$\frac{1}{6}$
3	$\frac{1}{6}$
4	$\frac{1}{6}$
5	$\frac{1}{6}$
6	$\frac{1}{6}$

Proces als kansverdeling

- Neem steekproeven van kansverdeling voor nieuwe uitkomsten
- Voor PCG hebben we alleen steekproeven nodig
- Generatieve functie met stochasten en parameters

Eigenschappen kansverdeling

- $p(X = x) \geq 0$ voor elke x
  - De kanswaarde van een uitkomst is nooit negatief
- $\sum_{x \in \mathcal{X}} p(X = x) = 1$
  - De som van alle kanswaarden is 1

Voorbeeld: textures genereren

- Afbeelding
  - 8bit pixels, 256 waarden, 3 kleuren
  - Resolutie van 16x16 pixels
- $256^{3 \cdot 16^2} \gg 10^1000$ mogelijke afbeeldingen
- Niet elke afbeelding is even “waarschijnlijk”

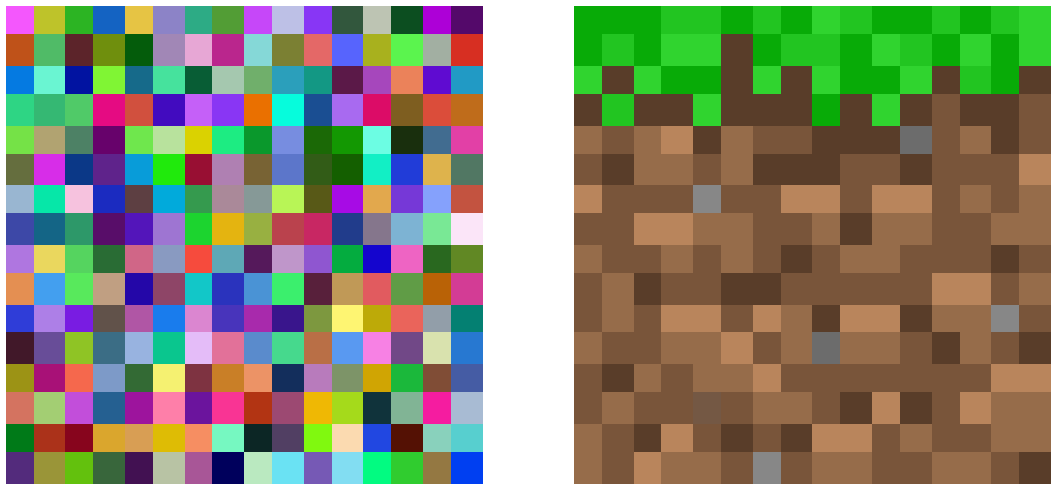


Figure 4: Welke afbeelding is door een texture artist gemaakt?

Het doel

- Schrijf een kanswaarde toe aan elke mogelijke uitkomst van een proces
- "Genereer" uitkomsten door steekproeven te nemen uit de kansverdeling
  - Waarschijnlijke/interessante uitkomsten zullen veel vaker verschijnen
- Kans van een uitkomst x is $p(X = x)$
- Als x een vector is:
  - $p(X = x) = p(X_1 = x_1, X_2 = x_2, \cdots, X_D = x_D)$
  - Simultane kansverdeling over alle componenten

Simultane kansverdelingen

- Als X en Y *onafhankelijk* zijn:
  - $p(X = x, Y = y) = p(X = x)p(Y = y)$
- Zelden waar voor interessante gevallen
  - Zoals textures
- Elke mogelijke kanswaarde moet worden berekend

Voorwaardelijke kansverdelingen

- Als de uitkomst van X afhankelijk is van de uitkomst van Y:
  - $p(X = x, Y = y) = p(X = x|Y = y)p(Y = y)$
- *Marginale* en *voorwaardelijke* kansverdelingen zijn vaak eenvoudiger dan *simultane* kansverdelingen

Voorwaardelijke pixels

- Zwart wit afbeelding van 16x16 pixels, 8 bits per pixel
- Meer dan 10^{617} uitkomsten
- Genereer elke pixel a.d.h.v. de vorige pixel
  - Kansverdeling over eerste pixel: 256 uitkomsten
  - Kansverdeling over pixel gegeven vorige pixel: 256 uitkomsten

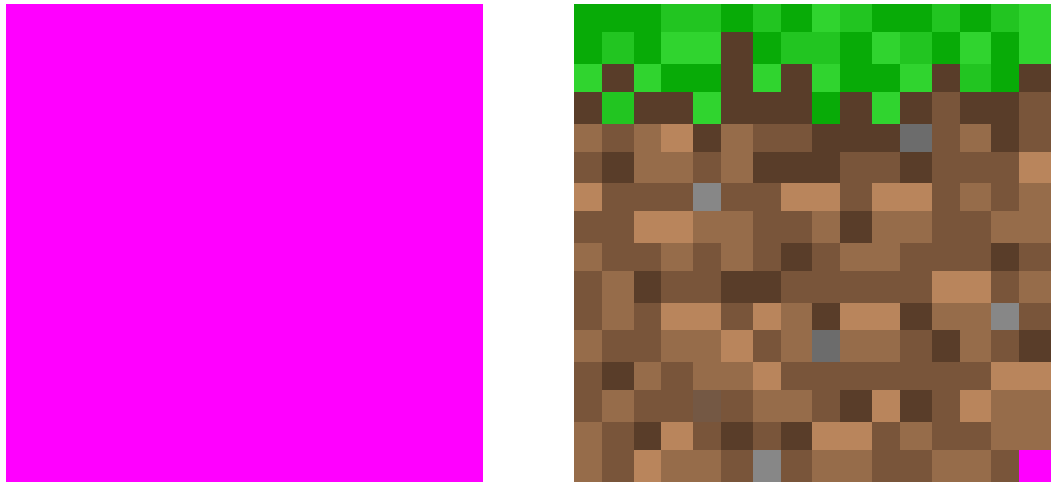


Figure 5: Welke kleur moet pixel (16, 16) hebben?

Tekst generatie

Reeksen

- Afbeeldingen zijn niet vanzelfsprekende reeksen
  - Links naar rechts of rechts naar links?
- Voor nu: verander niet de oplossing; verander het probleem
- Voorbeeld van data in reeksformaat: tekst

Generatief model voor tekst

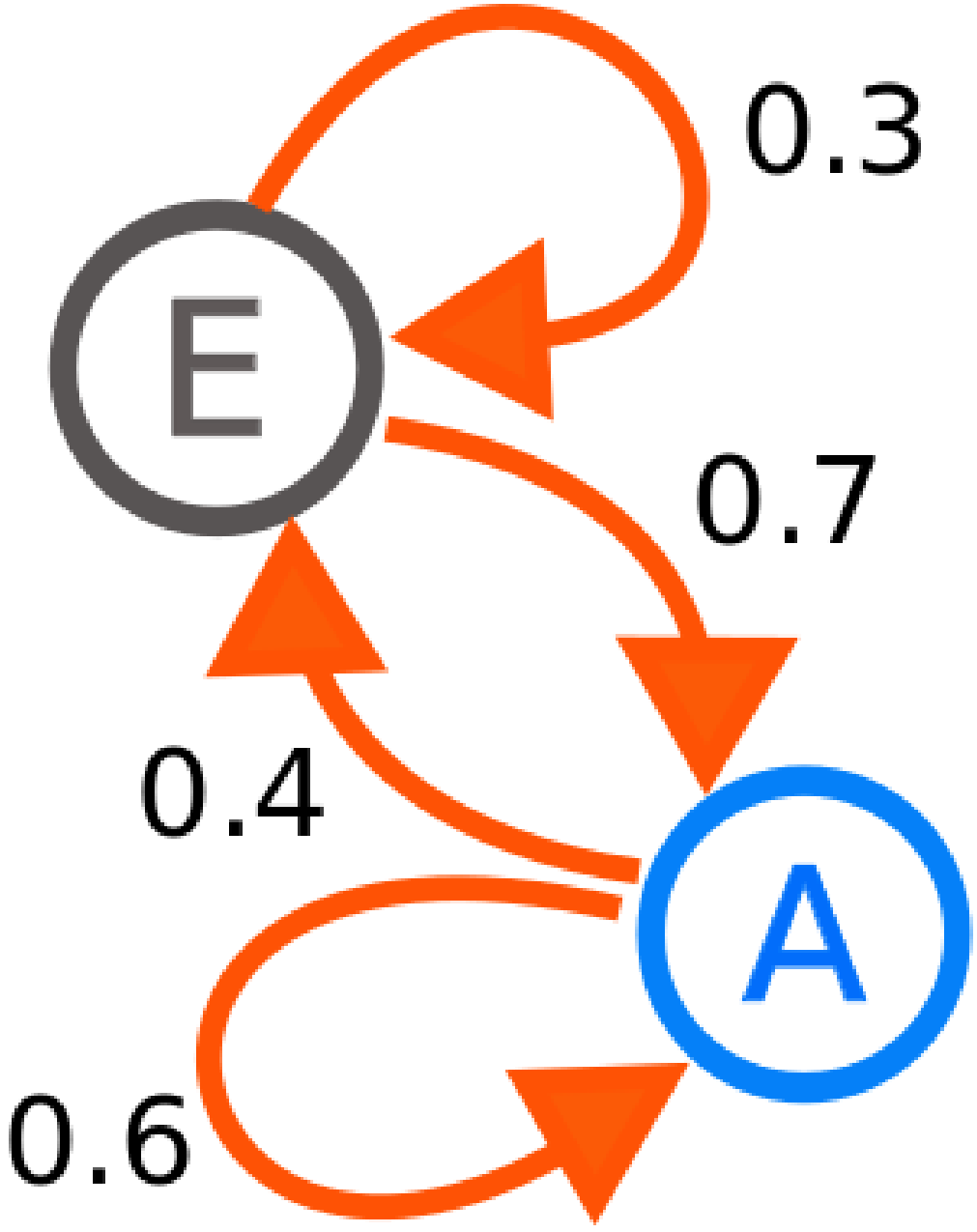
- $x="hallo"
- $p(X = x) = p("H", "a", "l", "l", "o")$
- De kans om een letter te observeren hangt af van eerdere letters

Markov keten

- Markov: “de toekomst is onafhankelijk van het verleden gegeven het heden”
- Voorspel de volgende letter door naar de vorige letter(s) te kijken
- Voorspel de volgende letter in deze reeks: “twaal”

Markov keten

- Stochastische *finite state machine*
- Bereken voor elke staat de kans om naar elke volgende staat over te gaan


0.3
E
0.7
0.4
A
0.6

Train een Markov keten

- Vind de kansverdeling $p(X = x|Y = y)$ voor elke x en y
- Schat de kans om letter x te zien gegeven dat we y zagen
- Tekst data: *corpus*

Voorbeeld

- Corpus: `"deze zin heeft vijf woorden"`
- Verzamel alle letterparen/*bigrams*:
  - `"de", "ez", "ze", "e_", ..., "en"`
- Tel hoe vaak we elke bigram tegenkomen:
  - `"de": 2, "ez": 1, ..., "en": 1`
- Normaliseer de tellingen:
  - `"ze": 1 / 2, "zi": 1 / 2`
  - Verzamel alle bigrams die met dezelfde letter *eindigen*
  - Deel elke telling door de som van alle tellingen

Genereer tekst

```
Input:
    - een dict met bigrams als key en de kanswaarde als val
    - k, aantal karakters om te genereren

result = "a"  # Begin met willekeurige letter

voor elke iteratie in 1..k:
    previous = laatste letter in resultaat

    bigrams = alle bigrams die beginnen met previous
    weights = alle kanswaarden die bij bigrams horen

    selection = random.choices(bigrams, p=weights)[0]

    # Alleen het laatste karakter van de bigram is relevant
    result = result + selection[-1]

Output: result
```

Resultaat

- Corpus: Roodkapje, `k=100`
- `"r agop gg, jnge n en ze scherten lededangraropr bien n, demoden zerde daan aap rm erd zon van ar olom"`
- Succes?

Ngrams

- 1 letter terug in de tijd is niet genoeg informatie
- We kunnen ook 2, 3, ... n letters terugkijken
  - Ngrams
- Er zijn meer combinaties van 3 letters dan van 2 letters
  - C^N, C is aantal karakters
  - Hoe groter N, hoe meer tekst we nodig hebben

Resultaat

- k=50, “Seed”: "roodkapje"
- N=3: “roodkapje meen rootmoe, waste bij sch was een en om isje no”
- N=5: “roodkapje, wat vind allebei pakker is nog een sprookjes val”
- N=8: “roodkapje had meegebracht, en die dacht: "wat vind ik je hi”

Langere tekst

"roodkapje had ondertussen een fles wijn, die roodkapje tegen haar bed liggen, sliep in en begon met een schaar de buik van de wolf af en trok hij haar bed liggen en toen kwam de oude vrouw; als je bij haar beter. maar recht aan naar haar toen hij voor het bed en at de grootmoeder, wat heb je grootmoeder, wat"

Conclusie

- Markov keten leert spelling, interpunctie, grammatica
  - Tot op zekere hoogte
- Markov eigenschap kan geen “lange afstandsrelaties” leren
- “De man die mij gisterochtend bij de supermarkt begroette was lang”
  - “De man” en “was lang” horen bij elkaar

Recurrente Neurale Netwerken

Neuraal netwerk als kansverdeling

- Gebruik een neuraal netwerk voor de functie $p(X = x|Y = y)$
  - Train op paren van inputs y en outputs x
- Verzeker dat de *totale* kanswaarde gelijk is aan 1
- Netwerk heeft een output voor elke uitkomst
  - Elk karakter
- Normaliseer alle outputs

Letters in een neuraal netwerk

- Neurale netwerken werken alleen met getallen
  - Wij gebruiken tekst karakters
- Simpele aanpak: een getal voor elke letter (a=1, b=2, etc.)
  - Voorspel “z” i.p.v. “s”; error is groter dan voor “y”
  - “fietz” is beter dan “fiety”
- Volgorde van encoding heeft invloed op resultaten
- Beschouw elke letter als eigen categorie

Categorische voorspelling

- Elk karakter is een vector
- Component voor elk karakter
- ELke waarde nul, behalve de waarde die met het karakter overeen komt
- One-hot vector
- Bijvoorbeeld; uitkomsten “a”, “b”, “c” en “d”
  - One-hot vector voor “b”: `[0, 1, 0, 0]`
- Gebruik one-hot vectors voor input en output

Softmax

- Normaliseer voorspelling neuraal netwerk met vier outputs
  - `z=[-2, 3, 0.4, 0.2]`
- $h(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
  - `np.exp(z) / np.sum(np.exp(z))`
  - Exponent maakt alle waardes positief
  - Deling door som zorgt dat alle waardes optellen tot 1
- Elke $h(z)_i$ geeft de kans voor de letter met index i

Markov keten met neuraal netwerk

- C karakters
- Input: vorige letter als one-hot vector (C-dimensionaal)
- Output: softmax (C-dimensionaal)
  - Kansverdeling over mogelijke volgende letters
- Voor ngrams: plak N one-hot vectoren aan elkaar (NC-dimensionaal)

Neuraal netwerk met geheugen

- "Verborgen" neuronen in neuraal netwerk hebben nuttige informatie over input
- Gebruik verborgen neuronen van vorige voorspelling als input voor volgende voorspelling
- Output wordt voorspelt a.d.h.v. huidige observatie en informatie over vorige observatie
- Gebruik zelfde parameters voor elke stap
  - Kan oneindig lange sequenties verwerken

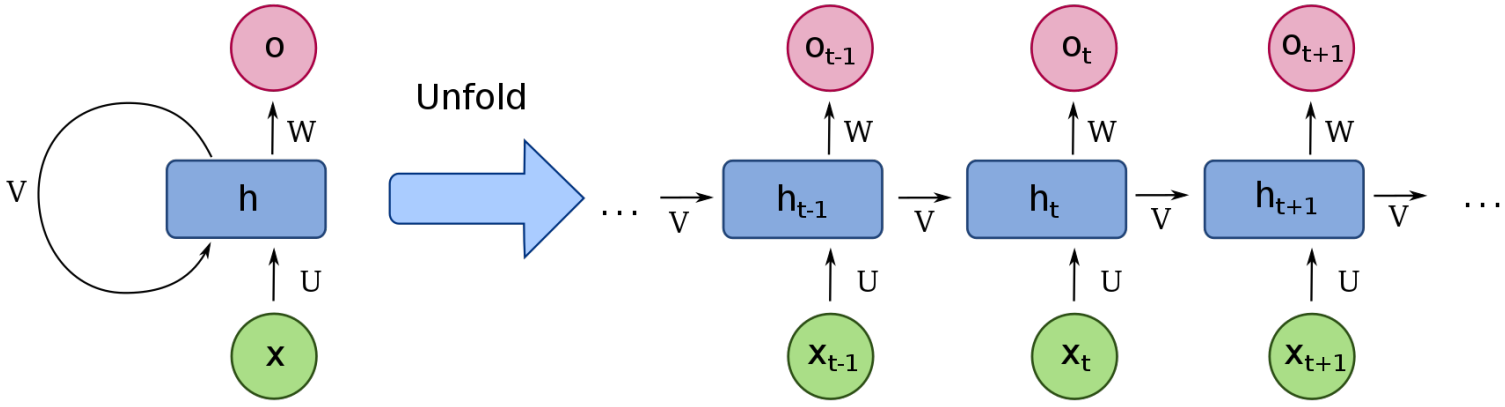


Figure 7: Recurrent netwerk

Vanishing/exploding gradients

- "Uitgerold" netwerk is eigenlijk een heel diep feed-forward netwerk
- Gradient van netwerk met respect tot parameters in de eerste laag
  - Gebruikt "kettingregel" met veel vermenigvuldigingen
- Vermenigvuldig alle waarden $x_1, x_2, \cdots, x_N$: $y = \prod_{n=1}^{N} x_n$
  - Als elke $x_n > 1$: y wordt heel groot
  - Als elke $x_n < 1$: y wordt heel klein

LSTM

- “Long short-term memory”
- Leer drie dingen
  - Welke informatie in huidige stap te onthouden
  - Welke informatie van vorige stap te vergeten
  - Welke informatie van vorige stap te gebruiken
- Kan veel langer informatie onthouden zonder vanishing/exploding gradients

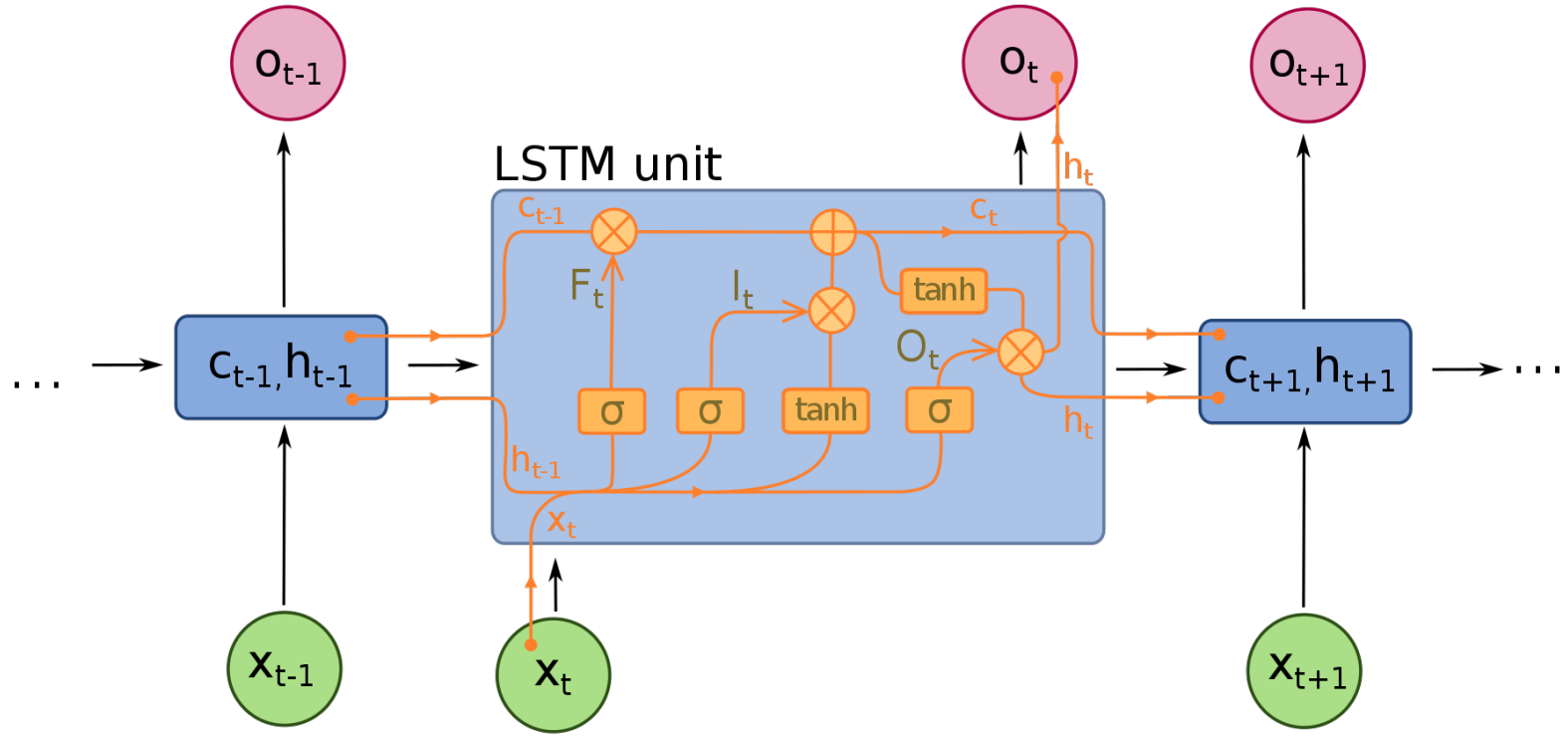


Figure 8: LSTM unit in recurrent netwerk

Tokens

- Trainen op letters: leer wat “woorden” zijn
  - Kost extra tijd en capaciteit
- Gebruik hele woorden
  - Maak een “woordenboek” met alle bekende woorden
  - Maak voor elk woord een one-hot vector met de indices van het woordenboek
- Extra “tokens” voor:
  - Onbekende woorden
  - Begin en eind van een zin/reeks
  - Speciale tekens (leestekens/emoji)
- Inputs worden groter, maar trainen wordt makkelijker

Toepassingen van recurrente netwerken

- Recurrente netwerken hebben inputs en outputs van variabele lengte
- Elke soort data met volgende eigenschappen
  - Reeks als input, voorspelling als output
  - Observatie als input, reeks als output
  - Reeks als input, reeks als output

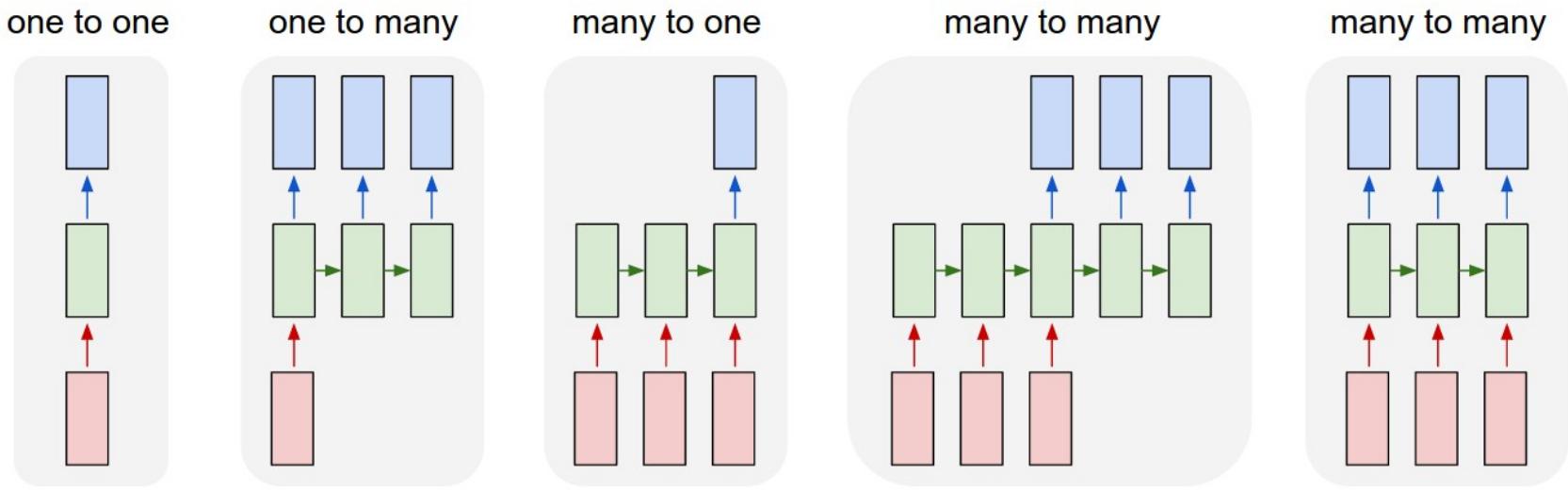


Figure 9: Toepassingen van RNNs

Reeksen data

- Veel data komen in de vorm van reeksen
  - Tekst, audio, video
- RNNs kunnen functies tussen verschillende domeinen van reeksen leren
  - Bijvoorbeeld: praak-naar-tekst

Afbeeldingen als reeksen

- Tekening: reeks van lijnen
- Elke lijn is een reeks van punten
- sketch-rnn