Algorithmische Komposition mit Neuronalen Netzen

AICA Crashkurs

Benedikt Zönnchen

17. Juni 2023

Inhalt

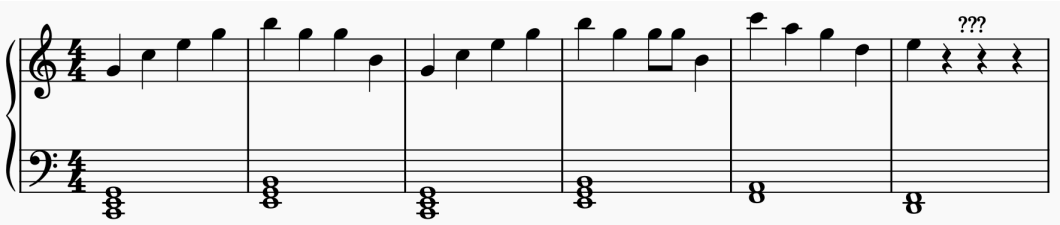
"[The mechanical engine] might act upon other things besides numbers, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine. [...] Supposing, for instance, that the fundamental relations of pitched sounds in the science of harmony and of musical composition were susceptible of such expression and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent." – Ada Lovelace (1815 - 1852)

Einleitung

Einleitung: Problemdefinition

Problem

Gegeben sei eine Sequenz von Noten der Länge $n \geq 0$. Generiere eine “wohlklingende” Fortsetzung.



Einleitung

Die Generierung guter Melodien und Harmonien ist ein schwieriges Problem:

- Mehrdimensionale Beziehungen (vertikale Harmonie und horizontale Melodie)
- Relative Beziehungen (Intervalle)
- Hoch- und niederfrequente Beziehungen
- Teils einzigartige Stücke
- Spannung zwischen Wiederholung und Überraschung
- ...

Anders als Sprache ist Musik ein Spiel mit Erwartungen – ein Zusammenspiel aus Wiederholung und Überraschung.

Die Beziehung zwischen symbolischer Notation und Ton ist ähnlich zur Beziehung zwischen Text und Sprache:

- **Partitur**: Symbolisch, abstrakt, transportiert musikalische Gedanken
- **Ton**: Durchgängig, konkret, enthält alle Details

Zwischen diesen beiden Extremen liegen oftmals weitere Schritte von abstrakt zu konkret.

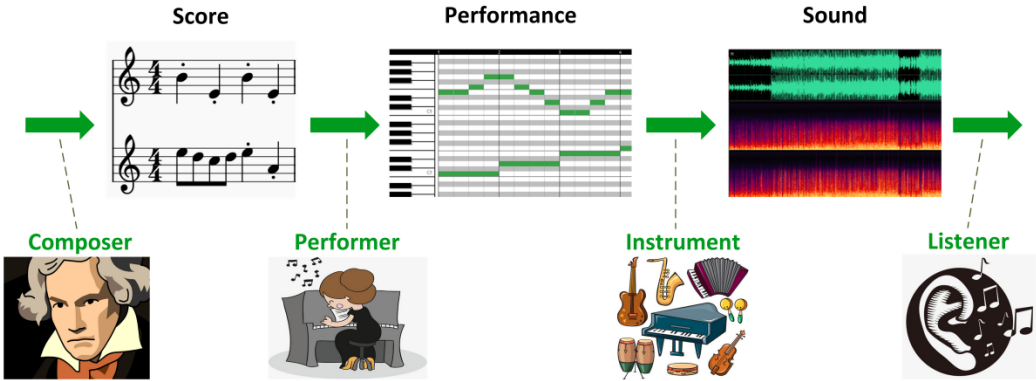


Abbildung: Quelle: [Ji et al., 2020]

Nach [Nierhaus, 2009] können wir zwischen den Extremen

- **authentische Komposition** und
- **Imitation** (des Stils),

unterscheiden, wobei wir uns immer zwischen den Extremen befinden werden.

Zusätzlich können wir zwischen weiteren Extremen unterscheiden, nämlich

- **regelbasierte** und
- **datengetriebene**,

sowie

- **automatisierte** und
- **manuelle** Komposition.

Regelbasierte Techniken

- **Markov-Chain (MC)**
- Hidden Markov Model (HMM)
- Zelluläre Automaten
- Generierende Grammatiken
- Lindenmayer-Systeme
- Fraktale
- Genetische Algorithmen
- ...

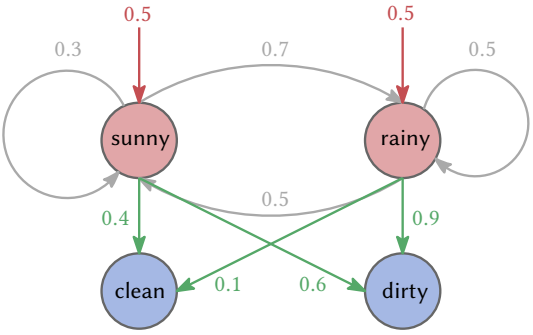


Abbildung: Einfaches Hidden Markov Model.

Datengetriebene Techniken

- **Markov-Chain (MC)**
- Hidden Markov Model (HMM)
- Neuronale Netzwerke:
  - **Feedforward Neuronal Networks (FNN)**
  - **Recurrent Neuronal Networks (RNN)**
  - Transformer
  - ...
- ...

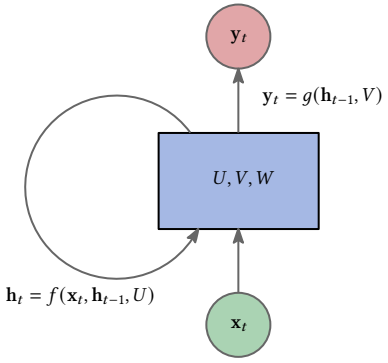


Abbildung: Skizze eines RNN.

Regelbasierte Kompositionen:

- Guido von Arezzo (1000): Generierung von Melodien aus Text
- Atanasius Kirchner (1650): Automatische Erzeugung von kontrapunktischer Komposition durch hölzerne Stöcke
- Würfelspiele von J. K. Kirnberger (1757 - 1812) und anderen wie Haydn, Mozart
- L. Hiller und L. Isaacson (1955): *The Iliac Suite*, eine vollständig computergenerierte Komposition basierend auf Markov-Ketten.
- Iannis Xenakis (1922 - 2001) z.B. *Metastasis*, John Cage (1912 - 1992) z.B. *Atlas Eclipticalis*: Zufallsbasierte Kompositionen

Datengetriebene Kompositionen:

- Vanilla Recurrent Neural Network (Einstimmig), [Todd, 1989]
- LSTMs für Melodie und Harmonie, [Eck and Schmidhuber, 2002]
- FolkRNN, LSTM (Einstimmig), [Sturm et al., 2016]
- LSTM mit einer interessanten Architektur (Vielstimmig), [Johnson, 2017]
- MusicTransformer, Komposition langer Sequenzen mit erhöhter Dynamik (Vielstimmig), [Huang et al., 2018]
- AnticipateRNN, bedingte Melodiegenerierung (Vielstimmig), [Hadjeres and Nielsen, 2018]
- Transformer, bedingte Melodiegenerierung (Vielstimmig), [Hadjeres and Crestel, 2021]

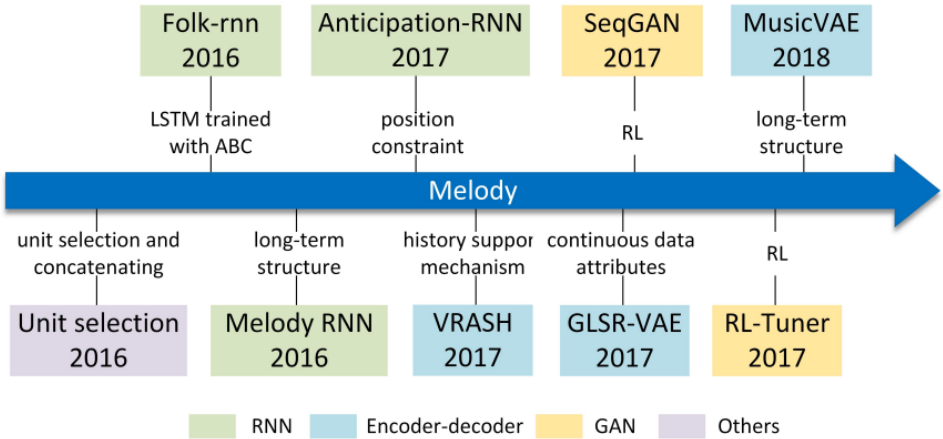


Abbildung: Quelle: [Ji et al., 2020]

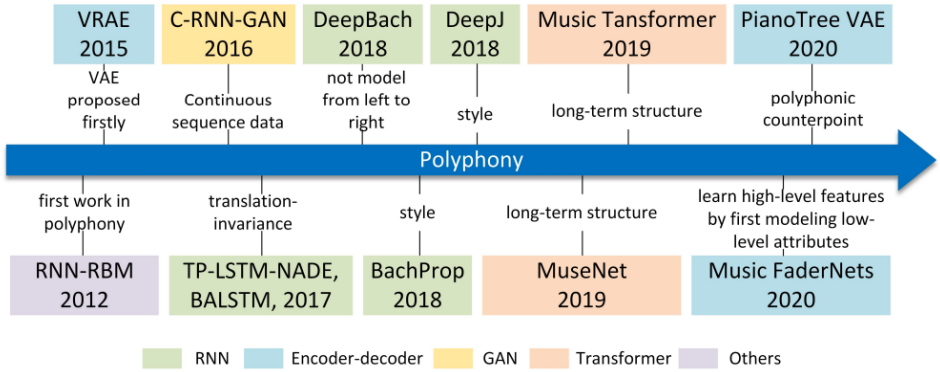


Abbildung: Quelle: [Ji et al., 2020]

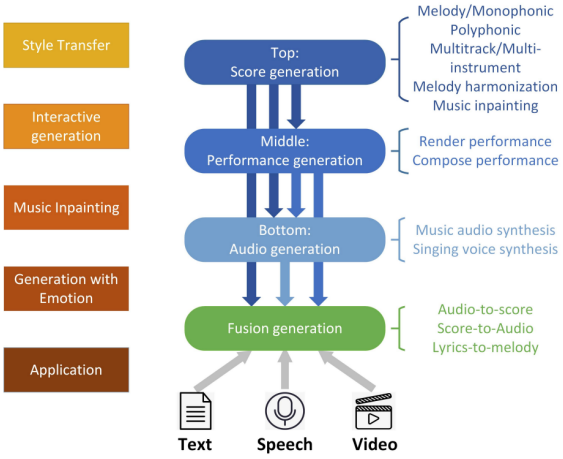


Abbildung: Quelle: [Ji et al., 2020]

Abgrenzung

Wir beschäftigen uns mit der *abstrakten* Melodiegenerierung (ohne Harmonie).

Repräsentationen

Repräsentationen

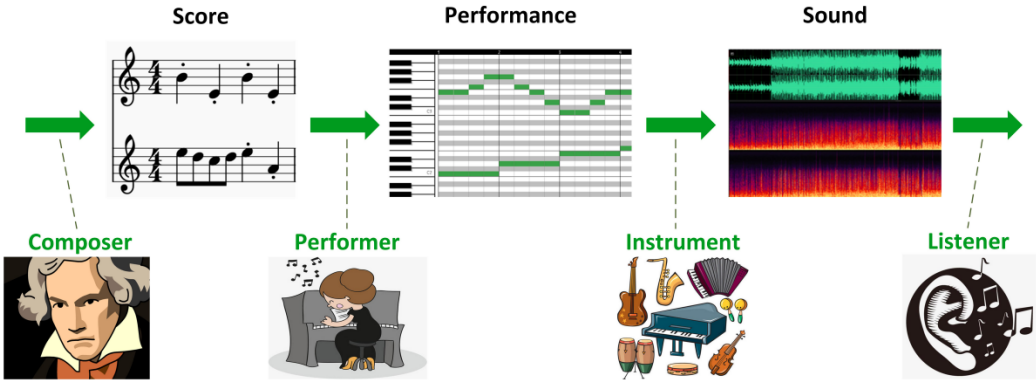


Abbildung: Quelle: [Ji et al., 2020]

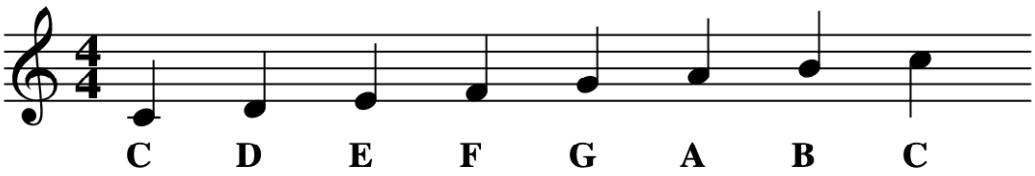
Repräsentationen

Wir arbeiten mit zwei Repräsentationen:

(MIDI-)Notensequenz

Jedes Ereignis ist eine Note der Form: [Tonhöhe in MIDI]/[Dauer in Vielfache von Zeitschritt]. Zum Beispiel entspricht 60/3 einem C3 der Länge 3/8 (Beats), wenn ein Zeitschritt 1/8 Beat wäre. Eine Melodie sieht dann wie folgt aus:

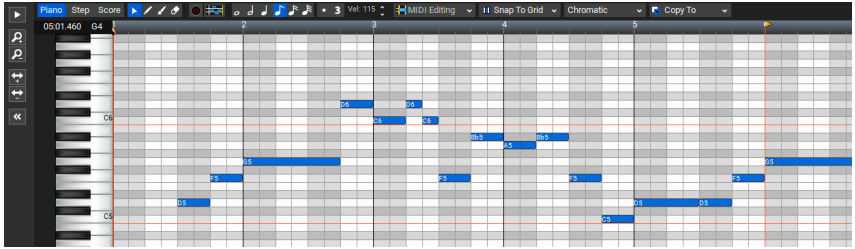
64/4 55/1 53/2 . . .



Piano Roll

Jedes Ereignis nimmt die gleiche Zeit in Anspruch. Es gibt für jede Note ein “Note wurde geschlagen”. Zudem gibt es ein Ereignis “Note/Pause wird gehalten” und ein Ereignis “Pause”. Eine Melodie sieht dann wie folgt aus:

64 _ _ _ 55 53 _ . . .



Repräsentationen

Die **Notensequenz** benötigt ein größeres Alphabet. **Piano Roll** verwendet dafür mehr Symbole für die gleiche Information.

Im Falle der Vielstimmigkeit explodiert unser Alphabet!

Frage:

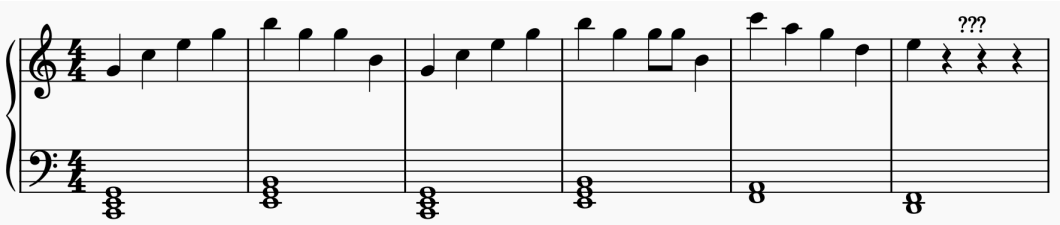
Wie könnte unsere Repräsentation im Falle der Vielstimmigkeit aussehen?

Repräsentationen

(Datengetriebene) Markov-Ketten

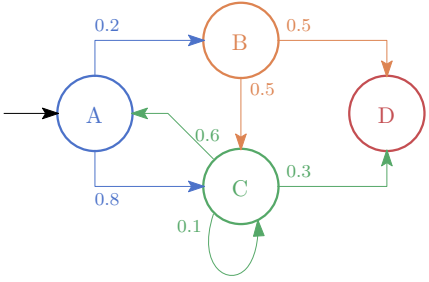
Problem

Gegeben sei eine Sequenz von Noten der Länge $n \geq 0$, generiere “wohlkingende” Fortsetzung.



Lösungsidee

Fasse jede Note als Zustand eines Automaten auf und berechne die **Übergangswahrscheinlichkeiten** anhand der gegebenen Daten.



Für eine solche Markov-Kette erster Ordnung hängt der nächste Zustand ausschließlich vom vorherigen ab!

Markov-Ketten

Angenommen wir haben m verschiedene Noten, dann gibt es m^2 Übergänge (manche haben möglicherweise eine Wahrscheinlichkeit von 0).

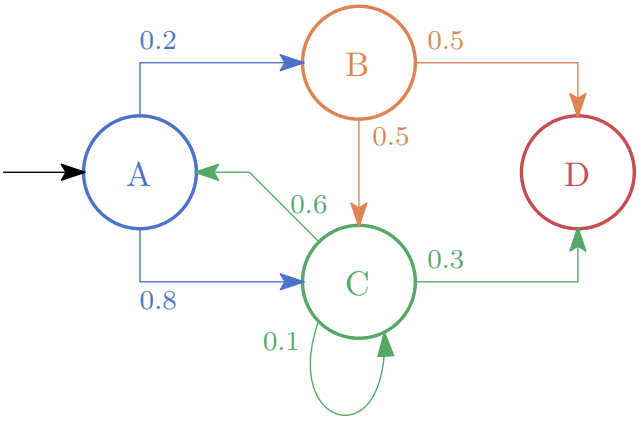
Wir gehen über alle Daten und zählen wie oft welcher Übergang auftritt. Anschließend teilen wir das Ergebnis durch die Anzahl der gesamten Übergänge.

Wir erzeugen demnach eine Tabelle oder Matrix $\mathbf{P}$ deren Zeile j, die Wahrscheinlichkeitsverteilung für eine die Note j angibt.

$$\mathbf{p}_j = (0.6, 0.0, 0.1, 0.3)$$

Eintrag i in Zeile j gibt demnach die Wahrscheinlichkeit für den Übergang der Note j nach i an.

Markov-Ketten



	A	B	C	D
A	0.0	0.2	0.8	0.0
B	0.0	0.0	0.5	0.5
C	0.6	0.0	0.1	0.3
D	0.0	0.0	0.0	0.0

$$\mathbf{P} = \begin{pmatrix} 0.0 & 0.2 & 0.8 & 0.0 \\ 0.0 & 0.0 & 0.5 & 0.5 \\ 0.6 & 0.0 & 0.1 & 0.3 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix}$$

Angenommen wir hätten folgendes **(MIDI-)Notensequenz**

. 64/4 64/2 50/3 50/3 64/4 64/2 .

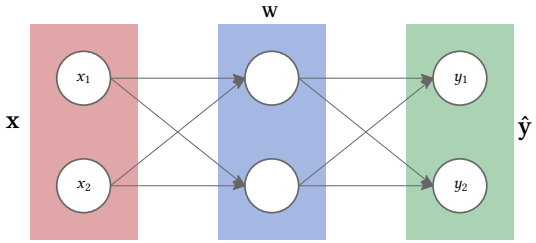
d.h., wir hätten 3 unterschiedliche Noten und ein spezielles Symbol: **50/3**, **64/2**, **64/4**, **.**
Zählen wir nun die Übergänge:

	50/3	62/2	64/4	.
50/3	1	0	1	0
62/2	1	0	0	1
64/4	0	2	0	0
.	0	0	1	0

Feedforward Neural Networks

Lösungsidee

Trainiere ein einfaches neuronales Netz, welches als Eingabe eine Sequenz mit k Noten erhält und die nächste Note vorhersagen soll.



Wir verwenden im Beispielnotebook $k = 1$ und verzichten auf Aktivierungsfunktionen. Damit ist unser Netzwerk eine einfache $m \times m$ Matrix $\mathbf{W}$.

Feedforward Neural Networks

Jede der m Noten kodieren wir mit einem sog. *one-hot Vektor* mit m Komponenten, wobei genau eine davon gleich 1 ist, z.B.

$$\mathbf{x}^\top = (0, 0, 1, \ldots, 0, 0, 0)$$

und die nächste Note $\hat{\mathbf{y}}$ ergibt sich aus

$$\mathbf{x}^\top \mathbf{W} = \hat{\mathbf{y}}$$

und wird als Wahrscheinlichkeitsverteilung über die m Noten interpretiert.

Wir trainieren unser FNN mit Sequenzen der Länge 1. D.h. aus einem Musikstück in **(MIDI-)Notensequenz**-Notation

. 64/4 64/2 50/3 50/3 64/4 64/2 .

erzeugen wir Trainingsdaten:

$$
\begin{array}{ccc}
. & \Rightarrow & 64/4 \\
64/4 & \Rightarrow & 64/2 \\
64/2 & \Rightarrow & 50/3 \\
50/3 & \Rightarrow & 50/3 \\
50/3 & \Rightarrow & 64/4 \\
64/4 & \Rightarrow & 64/2 \\
64/2 & \Rightarrow & . \\
\end{array}
$$

Recurrent Neural Networks

Recurrent Neural Networks

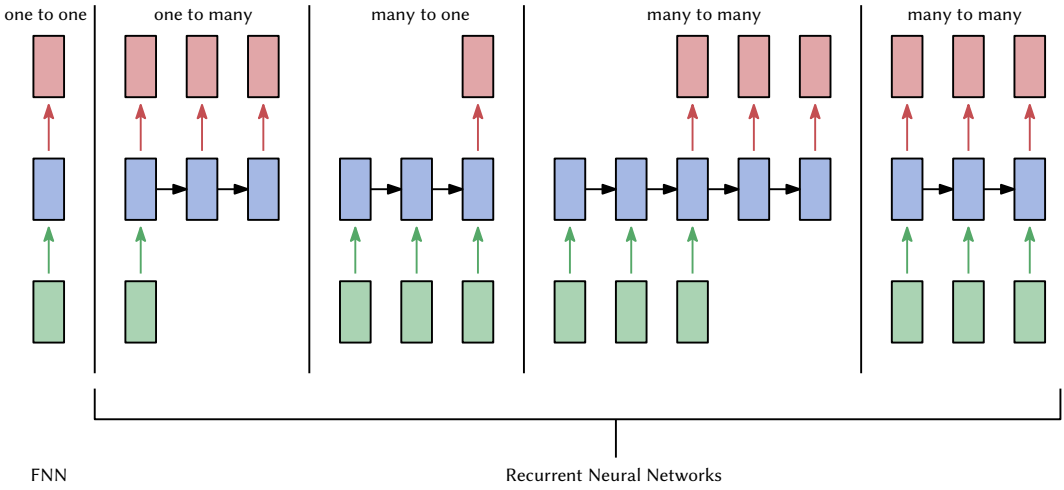
Problem

In beiden Ansätzen hängt die nächste Note lediglich von der aktuellen Note ab!

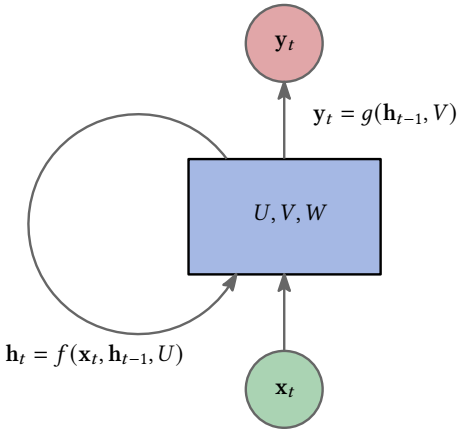
Mögliche Verbesserungen

- Markov-Ketten höherer Ordnung
- Transformer (vgl. erste Woche) aka “Lasse Netzwerk lange Sequenzen als Ganzes betrachten”
- **Recurrent Neural Networks** (“Netzwerk mit Speicher”)
- . . .

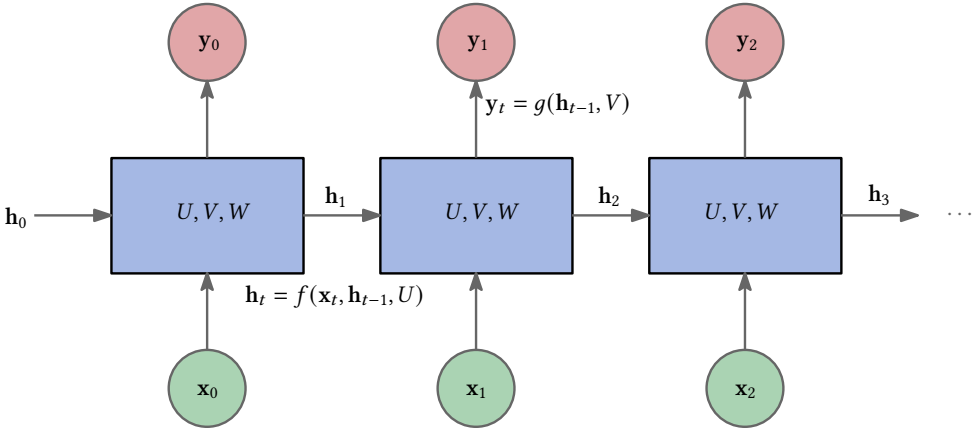
Recurrent Neural Networks



Recurrent Neural Networks



Recurrent Neural Networks

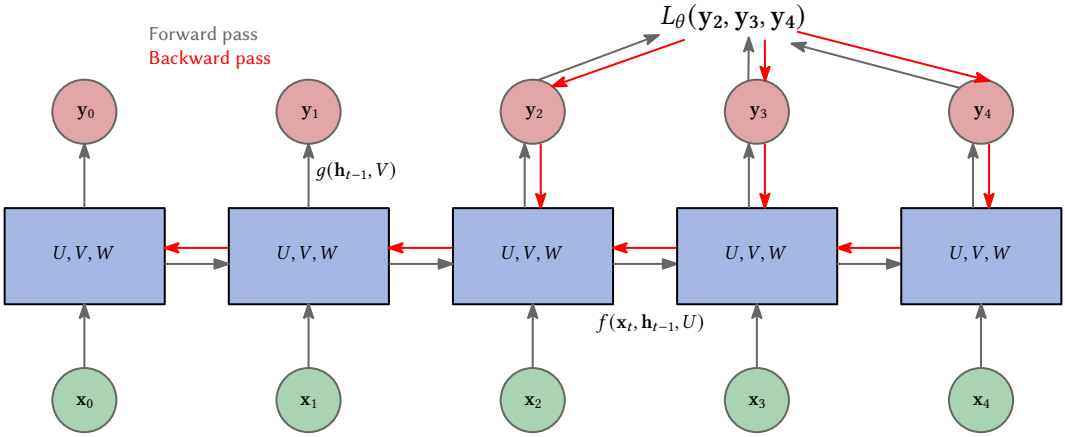


Theorem

*Für jedes **RNN** gibt es ein **Feedforward Network** mit dem gleichen Verhalten über eine endliche Zeit [Rumelhart and McClelland, 1987].*

Recurrent Neural Networks

Back-Propagation Through Time (BPTT)



"If training vanilla neural nets is optimization over functions, training recurrent nets is optimization over programs." – Andrej Karpathy

Es besteht eine exponentielle Abhängigkeit zwischen Fehler und Gewichten W

$$\frac{\partial \mathbf{h}_t}{\partial W} = \frac{\partial f(\mathbf{x}_t, \mathbf{h}_{t-1}, W)}{\partial W} + \frac{\partial f(\mathbf{x}_t, \mathbf{h}_{t-1}, W)}{\partial \mathbf{h}_{t-1}} \cdot \frac{\partial \mathbf{h}_{t-1}}{\partial W}$$

$\Rightarrow$ **Gradienten tendieren zu explodieren oder zu verschwinden.**

$$\lim_{N \to \infty} (x \cdot w_{ij}^N) = \begin{cases} \infty & \text{für } w_{ij} > 1.0 \\ 0 & \text{für } w_{ij} < 1.0 \end{cases}$$

Recurrent Neural Networks

Vanilla RNNs (in ihrer ursprünglichen Form) werden kaum noch benutzt. Stattdessen setzt man auf **Long Short-term Memory RNNs** kurz **LSTMs**.

Recurrent Neural Networks

Vorteile

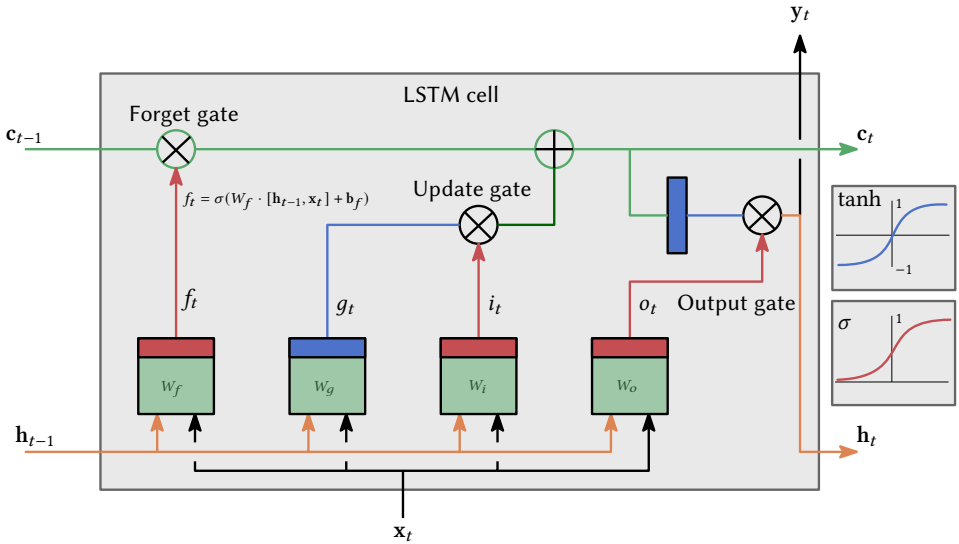
+ Variable Eingabegröße
+ Modellgröße explodiert nicht bei längerer Eingabe
+ Berechnung nimmt historische Informationen auf
+ Gewichte werden über die Zeit geteilt

Nachteile

- Langsame (sequenzielle) Berechnung
- Informationen die lange zurück liegen gehen verloren
- Keine Möglichkeit zukünftige Eingabe in die derzeitige Berechnung einzubinden

Recurrent Neural Networks

Long Short-term Memory



Vorteile (gegenüber Vanilla RNN)

+ Kommt mit längeren Abhängigkeiten klar
+ Stabilerer “Speicher”
+ Additiver statt Multiplikativer Fehler beim BPTT

Nachteile (gegenüber Vanilla RNN)

- Längeres Training mit mehr Ressourcen
- Anfällig für Überanpassung

Die sequenzielle Struktur, d.h. der **Informationsfluss durch die Zeit** bleibt bestehen!

$\Rightarrow$ Die Information wird weiterhin mit der Zeit verwischt.

Nehmen wir an, wir trainieren unser RNN mit Sequenzen der Länge 4. D.h. aus einem Musikstück in **Piano Roll**-Notation

64 _ _ _ 55 53 _ 63 _ _ _ r _ _ _ 55 _ _ _ 56 _ _ _

erzeugen wir Trainingsdaten:

.	.	.	.	⇒	64
.	.	.	65	⇒	_
.	.	65	_	⇒	_
.	65	_	_	⇒	_
65	_	_	_	⇒	55
_	_	_	55	⇒	53
			...		

			...		
_	56	_	_	⇒	_
56	_	_	_	⇒	.
_	_	_	.	⇒	.
_	_	.	.	⇒	.
_	.	.	.	⇒	.

In unserer Repräsentation erfassen wir lediglich die Information Tonhöhe und Tondauer mit:

64 _ _ _ 55 53 _ 63 _ _ _ r _ _ _ 55 _ _ _ 56 _ _ _

Welche Informationen wären möglicherweise noch hilfreich?

Referenzen I

Eck, D. and Schmidhuber, J. (2002).
Finding temporal structure in music: blues improvisation with LSTM recurrent networks.
In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, pages 747–756.

Hadjeres, G. and Crestel, L. (2021).
The piano inpainting application.
CoRR, abs/2107.05944.

Hadjeres, G. and Nielsen, F. (2018).
Anticipation-RNN: enforcing unary constraints in sequence generation, with application to interactive music generation.
Neural Computing and Applications.

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Hawthorne, C., Dai, A. M., Hoffman, M. D., and Eck, D. (2018).
Music transformer: Generating music with long-term structure.
arXiv preprint arXiv:1809.04281.

Ji, S., Luo, J., and Yang, X. (2020).
A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions.
CoRR, abs/2011.06801.

Referenzen II


Johnson, D. D. (2017).
Generating polyphonic music using tied parallel networks.
In *EvoMUSART*.


Nierhaus, G. (2009).
Algorithmic Composition - Paradigms of Automated Music Generation.
SpringerWienNewYork.


Rumelhart, D. E. and McClelland, J. L. (1987).
Learning Internal Representations by Error Propagation, pages 318–362.


Sturm, B. L., Santos, J. F., Ben-Tal, O., and Korshunova, I. (2016).
Music transcription modelling and composition using deep learning.
CoRR, abs/1604.08723.


Todd, P. M. (1989).
A Connectionist Approach To Algorithmic Composition.
Computer Music Journal, 13:27–43.