

Missing Data Imputation mit Variational Autoencodern

Niklas Brunn, Ben Deitmar, Sebastian Hahn
Jannis Klingler, Clemens Schächter

08.02.2021



Beispiel *time series*



Modellannahmen

Gegeben:

- Datensatz $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^{D \times N}$
- Datenpunkte $\mathbf{x}_i \sim p(\mathbf{x}_i)$ (u.i.v. Realisierungen einer hochdimensionalen Zufallsvariable)

Latente Variablen Modell

Suchen:

- Verteilung der Datenpunkte \mathbf{x}_i
- Eine niedrigdimensionale Darstellung der Daten, die den Prozess, der die Daten erzeugt, möglichst gut modelliert
- Führen dazu *Latente Variablen* \mathbf{z} , die auf dem *latenten Raum* $\mathcal{Z} = \mathbb{R}^d$ mit $d \ll D$ definiert sind, ein

Latente Variablen Modell

Das *Latente Variablen Modell* besteht nun aus den Verteilungen:

- Verteilung der Daten $p(\mathbf{x}_i)$
- Posteriorverteilung $p(\mathbf{z}|\mathbf{x}_i)$
- Priorverteilung $p(\mathbf{z})$
- Generatives Modell $p(\mathbf{x}_i|\mathbf{z})$

Latente Variablen Modell

Sind interessiert an der Verteilung

$$p(\mathbf{x}_i) = \int_{\mathbf{z}} p(\mathbf{x}_i, \mathbf{z}) d\mathbf{z} = \int_{\mathbf{z}} p(\mathbf{x}_i | \mathbf{z}) p(\mathbf{z}) d\mathbf{z}.$$

Problem: Dieses Integral kann aufgrund der hohen Dimension des Raumes \mathcal{X} und der Komplexität des Modells nicht berechnet und auch nicht sinnvoll approximiert werden.

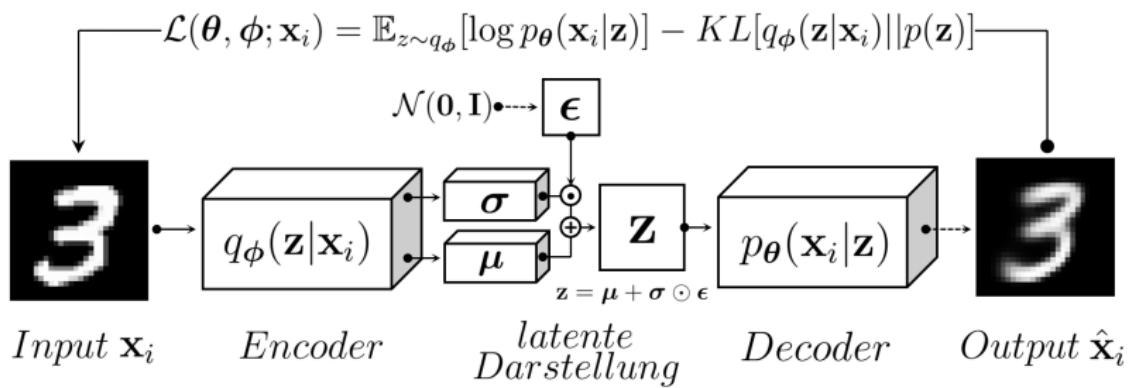
Mit dem Satz von Bayes gilt dies auch für die Verteilung

$$p(\mathbf{z} | \mathbf{x}_i) = \frac{p(\mathbf{x}_i | \mathbf{z}) p(\mathbf{z})}{p(\mathbf{x}_i)}.$$

Implementierung

- Führen nun das Inferenzmodell $q_\phi(z|x_i)$ ein und suchen die Parameter ϕ^* für die $q_{\phi^*}(z|x_i) \approx p_\theta(z|x_i)$ gilt
- In einem Variational Autoencoder werden die Verteilungen $q_\phi(z|x)$ und $p_\theta(x_i|z)$ durch neuronale Netze dargestellt und approximiert
- Können dann die Verteilung von $p(z)$ festlegen (häufig $\mathcal{N}(0, I)$)

Schematische Darstellung der VAE-Architektur



Verlustfunktion (ELBO)

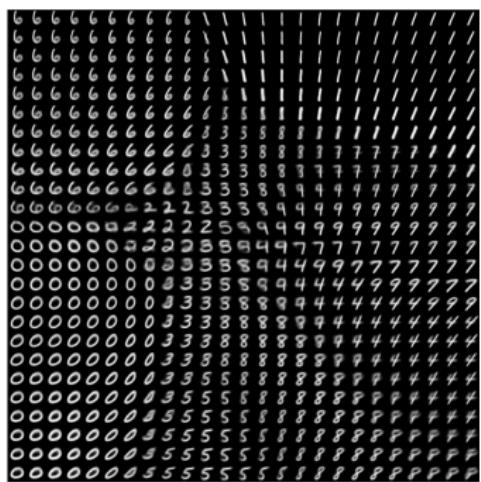
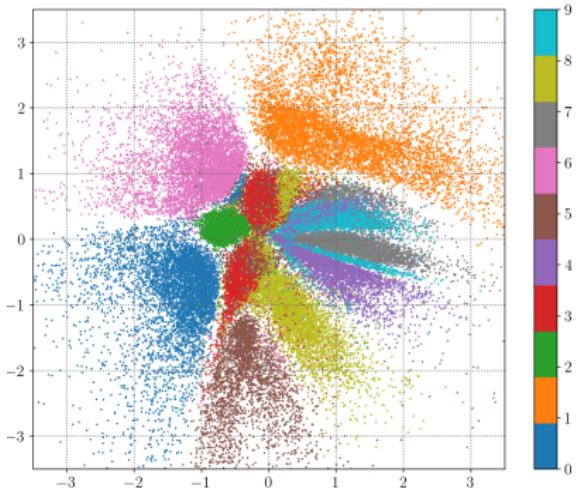
$$\log(p_{\theta}(\mathbf{x}_i)) - \underbrace{D_{KL}[q_{\phi}(\mathbf{z}|\mathbf{x}_i) || p_{\theta}(\mathbf{z}|\mathbf{x}_i)]}_{\geq 0}$$

$$= \underbrace{\mathbb{E}_{\mathbf{z} \sim q_{\phi}} [\log(p_{\theta}(\mathbf{x}_i|\mathbf{z}))] - D_{KL}[q_{\phi}(\mathbf{z}|\mathbf{x}_i) || p_{\theta}(\mathbf{z})]}_{=: \text{ELBO}} = -\mathcal{L}(\theta, \phi, \mathbf{x}_i)$$

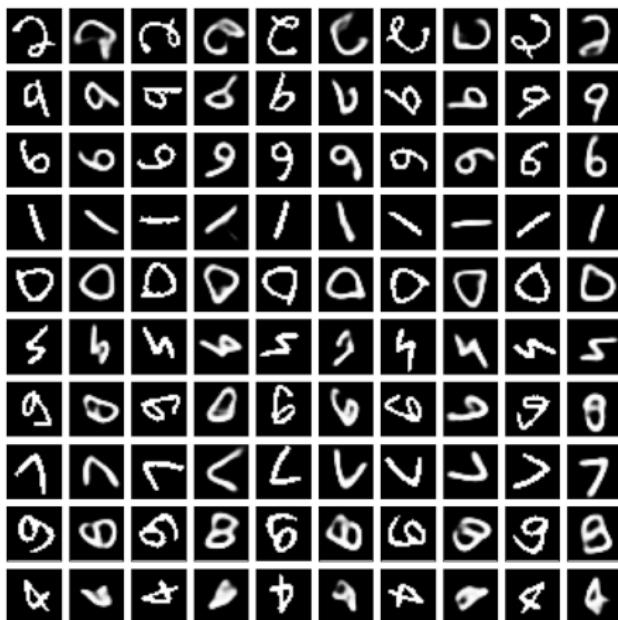
Verlustfunktion (ELBO)

$$\log(p_{\theta}(\mathbf{x}_i)) \geq \underbrace{\mathbb{E}_{\mathbf{z} \sim q_{\phi}} [\log(p_{\theta}(\mathbf{x}_i | \mathbf{z}))] - D_{KL}[q_{\phi}(\mathbf{z} | \mathbf{x}_i) || p_{\theta}(\mathbf{z})]}_{=: \text{ELBO} = -\mathcal{L}(\theta, \phi, \mathbf{x}_i)}$$

Beispiel eines latenten Raumes

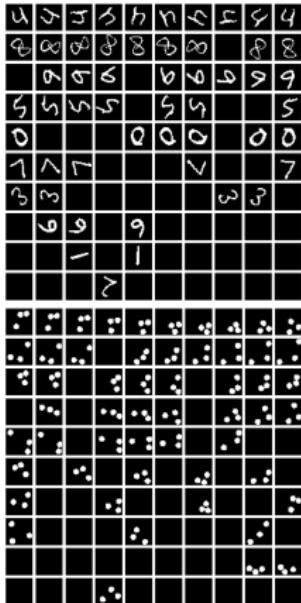


Interpolation im Latenten Raum

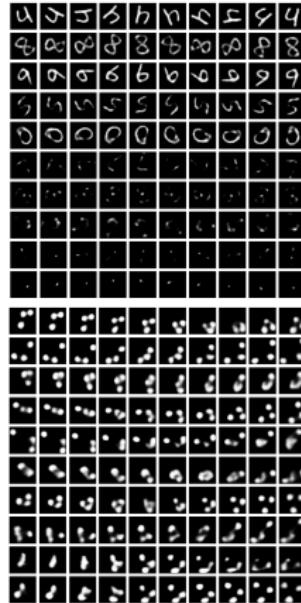


Vanilla-VAE

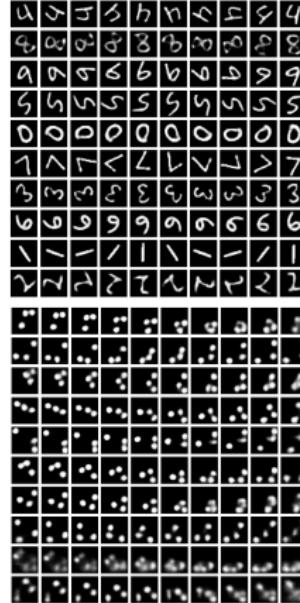
Modellinput



Trainingsdaten
vollständig



Trainingsdaten
beschädigt



ODE^2VAE

Wir wollen als nächstes *time series* mit einen ODE^2VAE modellieren

ODE^2VAE : Deep generative second order ODEs with Bayesian neural networks von Ç. Yıldız, M. Heinonen und H. Lähdesmäki

- Bewegungen (z.B. Fadenpendel) lassen sich nicht gut mit einem Standard-VAE modellieren
- **neuer Ansatz:** Zeitliche Abhangigkeit der Daten berucksichtigen und die Bewegung durch Differentialgleichungen zweiter Ordnung (ODE²) modellieren
- **Vorteile:** Wir konnen so zu beliebigen Zeitpunkten abfragen und auch Prognosen fur die Zukunft generieren

Notation

- **Gegeben:** time series $\mathbf{x}_{0:T} = (\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_T) \in \mathbb{R}^{D \times T}$ zu beobachteten Zeitpunkten t_0, t_1, \dots, t_T
- **Annahme:** Zeitliche Änderung der latenten Darstellung kann stetig durch eine ODE² beschrieben werden
- Zugehörigen latenten Zustände zu den beobachteten Zeitpunkten sind $\mathbf{z}_{0:T} = (\mathbf{z}_0, \mathbf{z}_1, \dots, \mathbf{z}_T) \in \mathbb{R}^{d \times T}$
- Latente Zustände werden in Positions- und Geschwindigkeitskomponente aufgeteilt $\mathbf{z}_t = (\mathbf{s}_t, \mathbf{v}_t)$

Notation

- Die ODE² lässt sich als ein System zweier ODE¹ schreiben

$$\frac{\partial^2 \mathbf{z}(t)}{\partial^2 t} = f_\psi \left(\mathbf{z}(t), \frac{\partial \mathbf{z}(t)}{\partial t}, t \right)$$

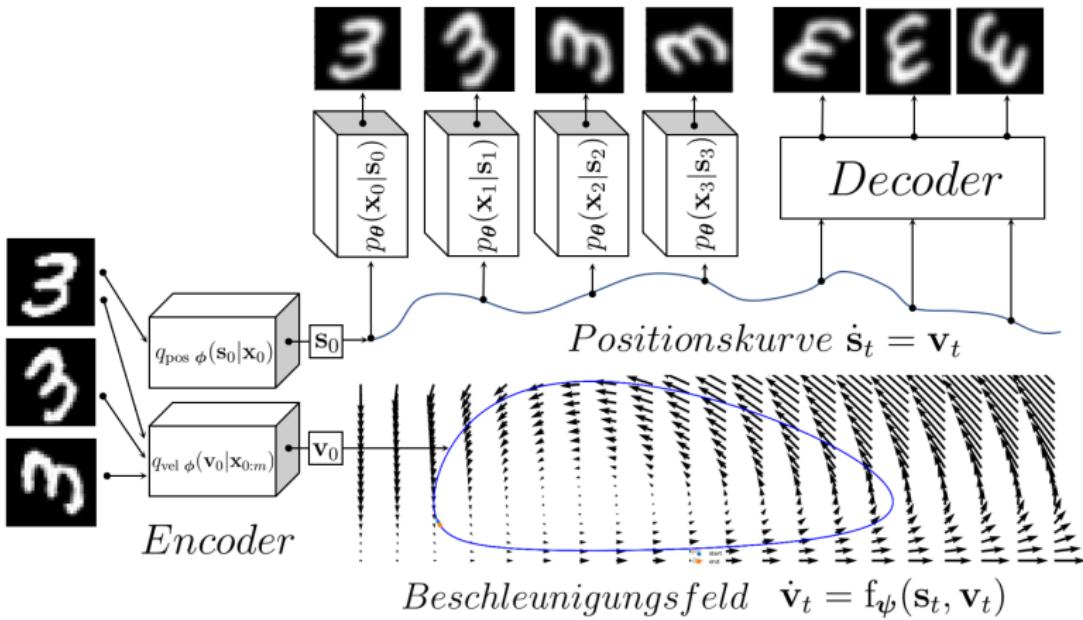
- f_ψ ist dabei ein NN, welches das Beschleunigungsfeld erlernt
- Lösen des Anfangswertproblems in $\mathbf{z}_0 = (\mathbf{s}_0, \mathbf{v}_0)$ liefert zu beliebigen Zeitpunkten T eine Lösung

$$\begin{pmatrix} \mathbf{s}_T \\ \mathbf{v}_T \end{pmatrix} = \begin{pmatrix} \mathbf{s}_0 \\ \mathbf{v}_0 \end{pmatrix} + \int_0^T \begin{pmatrix} \mathbf{v}(t) \\ f_\psi(\mathbf{s}(t), \mathbf{v}(t), t) \end{pmatrix} dt.$$

Das Modell besteht aus den folgenden Komponenten:

- Als **Prior** für die latente Ausgangsposition $p(\mathbf{s}_0)$ und die latente Ausgangsgeschwindigkeit $p(\mathbf{v}_0)$ wird eine Standardnormalverteilung gewählt
- Einem **Position-Encoder** welcher mit \mathbf{x}_0 die erste latente Ausgangsposition \mathbf{s}_0 encodiert und einem **Velocity-Encoder** welcher mit den ersten $m \geq 2$ Werten \mathbf{x}_m die latente Ausgangsgeschwindigkeit encodiert
- Einem **ODE-Netzwerk** welches zu einem Anfangswert eine Kurve im latenten Raum bestimmt
- Einem **Decoder**, welcher die aus der Kurve resultierenden latenten Positionen decodiert

Modell



Training

Als Trainingskriterium verwenden wir erneut die **ELBO**

$$\begin{aligned}\log(p_{\theta}(\mathbf{x}_{0:T})) &\geq \mathbb{E}_{\mathbf{z}_{0:T} \sim q_{\phi,\psi}} [\log(p_{\theta}(\mathbf{x}_{0:T}|\mathbf{z}_{0:T}))] - D_{KL}[q_{\phi,\psi}(\mathbf{z}_{0:T}|\mathbf{x}_{0:m})||p_{\theta,\psi}(\mathbf{z}_{0:T})] \\ &= \underbrace{\mathbb{E}_{\mathbf{z}_0 \sim q_{\text{enc } \phi}} [\log(p_{\theta}(\mathbf{x}_0|\mathbf{z}_0))]}_{\text{Vanilla-VAE ELBO}} - D_{KL}[q_{\text{enc } \phi}(\mathbf{z}_0|\mathbf{x}_{0:m})||p_{\theta}(\mathbf{z}_0)] \\ &\quad + \underbrace{\sum_{t=1}^T \mathbb{E}_{\mathbf{z}_t \sim q_{\text{oode } \psi}} [\log(p_{\theta}(\mathbf{x}_t|\mathbf{z}_t))]}_{\text{dynamic loss}} - D_{KL}[q_{\text{oode } \psi}(\mathbf{z}_t|\mathbf{x}_{0:m})||p_{\theta,\psi}(\mathbf{z}_t)]\end{aligned}$$

Während des Trainings sieht das Modell die gesamte *time series*

Ergebnisse

bouncing Balls



Ergebnisse

Von links nach rechts: Modellinput, tatsächlicher Verslauf der *time series*, Rekonstruktionen

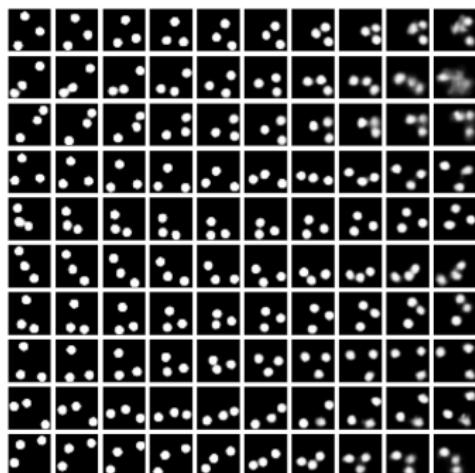
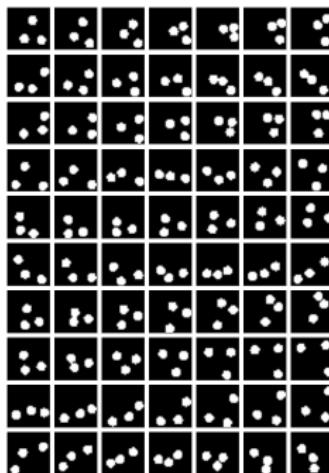
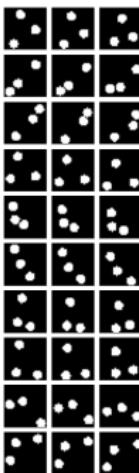
6	6	6
8	8	8
0	0	0
4	x	4
3	3	3
5	6	6
2	2	2
9	9	9
1	1	1
x	x	x

9	9	9	9	9	9	6	6
8	8	8	8	8	8	8	8
0	0	0	0	0	0	0	0
x	x	x	x	x	x	x	x
3	3	3	3	3	3	3	3
5	5	5	5	5	5	5	5
2	2	2	2	2	2	2	2
6	6	6	6	6	6	6	6
1	1	1	1	1	1	1	1
x	t	x	x	x	x	x	x

6	6	6	9	9	9	9	9	9	6	6
8	8	8	8	8	8	8	8	8	8	8
0	0	0	0	0	0	0	0	0	0	0
x	x	x	x	x	x	x	x	x	x	x
3	3	3	3	3	3	3	3	3	3	3
5	6	6	5	5	5	5	5	5	5	5
2	2	2	2	2	2	2	2	2	2	2
9	9	9	6	6	6	6	6	6	9	9
1	1	1	1	1	1	1	1	1	1	1
x	x	x	t	x	x	x	x	x	x	x

Ergebnisse

Von links nach rechts: Modellinput, tatsächlicher Verslauf der *time series*, Rekonstruktionen



Ergebnisse

AVI-Motion



Ergebnisse

Modellinput



Extrapolation



Erste Verallgemeinerung: ODE M-ter Ordnung

Bisherige ODE:

$$\nabla s(\cdot, \omega) = v(\cdot, \omega) \in \mathbb{R}^d$$

$$\nabla v(\cdot, \omega) = \mathbf{f}(s(\cdot, \omega), v(\cdot, \omega)) \in \mathbb{R}^d$$

Verallgemeinerung für $M = 3$:

$$\nabla s(\cdot, \omega) = \mathbf{f}_1(s(\cdot, \omega), v(\cdot, \omega), w(\cdot, \omega))$$

$$\nabla v(\cdot, \omega) = \mathbf{f}_2(s(\cdot, \omega), v(\cdot, \omega), w(\cdot, \omega))$$

$$\nabla w(\cdot, \omega) = \mathbf{f}_3(s(\cdot, \omega), v(\cdot, \omega), w(\cdot, \omega))$$

Schreibe $\mu = \mathbf{f} : \mathbb{R}^{M \times d} \rightarrow \mathbb{R}^{M \times d}$ und $z = (s, v, w)^T \in \mathbb{R}^{M \times d}$.

Zweite Verallgemeinerung: SDE

Bisherige ODE: $\forall k < M, \forall i \leq d :$

$$dz^{(k,i)}(t) = \mu^{k,i}(z(t)) dt$$

Verallgemeinerung für $\sigma : \mathbb{R}^{M \times d} \rightarrow \mathbb{R}^{M \times d \times n}$:

$$dz^{(k,i)}(t) = \mu^{k,i}(z(t)) dt + \sum_{j=1}^n \sigma^{k,i,j}(z^{(0)}(t), \dots, z^{(M-1)}(t)) dW_t^j$$

↳ ($M d$)-dimensionale SDE mit Brownschen Bewegungen (BBs)
 W^1, \dots, W^n .

BB als Random Walk

- Satz von Donsker:

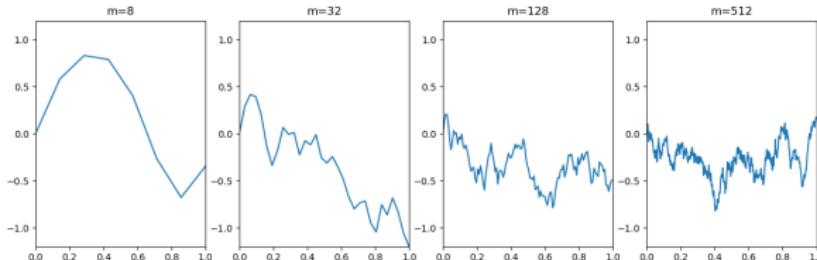
Seien Y_1, Y_2, \dots uiv. mit $\mathbb{E}[Y_i] = 0$ und $\sigma^2 := \mathbb{E}[Y_i^2] < \infty$.

Definiere den Prozess

$$W_{m,t} := \frac{1}{\sqrt{m}} \left(\sum_{i=1}^{\lfloor mt \rfloor} Y_i + (mt - \lfloor mt \rfloor) Y_{\lfloor mt \rfloor + 1} \right) \in C^0([0, \tau]).$$

Dann gilt $W_{m,\cdot} \xrightarrow{m \rightarrow \infty, \mathcal{L}} W$ (BB) in $C^0([0, \tau])$.

- Anschaulich:



SDE als ODE mit Random Walk

- eindimensionale SDE:

$$dX_t = \mu(t, X_t) dt + \sum_{j=1}^n \sigma^j(t, X_t) dW^j$$

- Ito-Diffusion (nicht Zeit-abhängig):

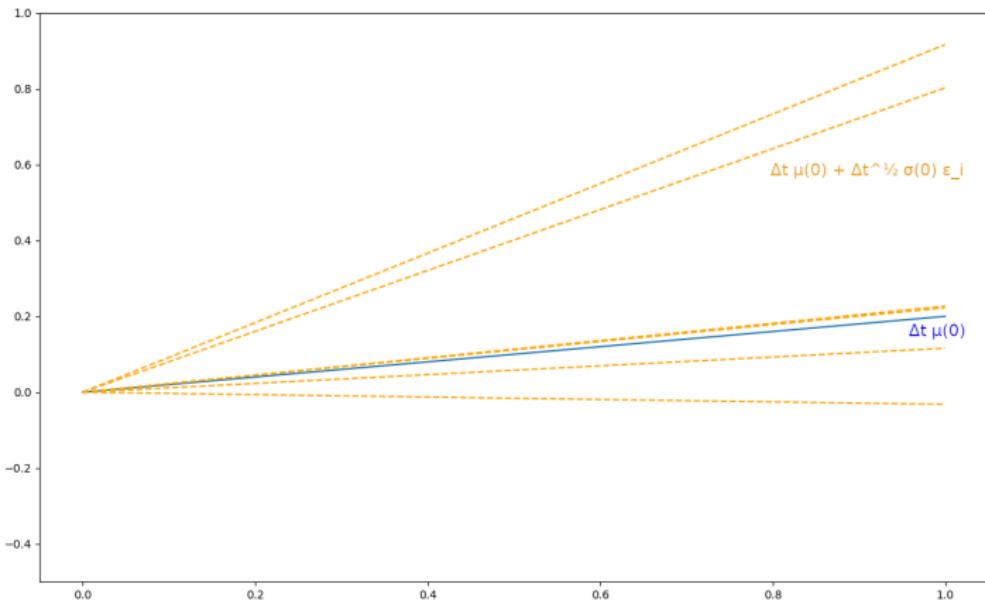
$$dX_t = \mu(X_t) dt + \sum_{j=1}^n \sigma^j(X_t) dW^j$$

- Euler-Maruyama-Verfahren um Lösung X zu approximieren:

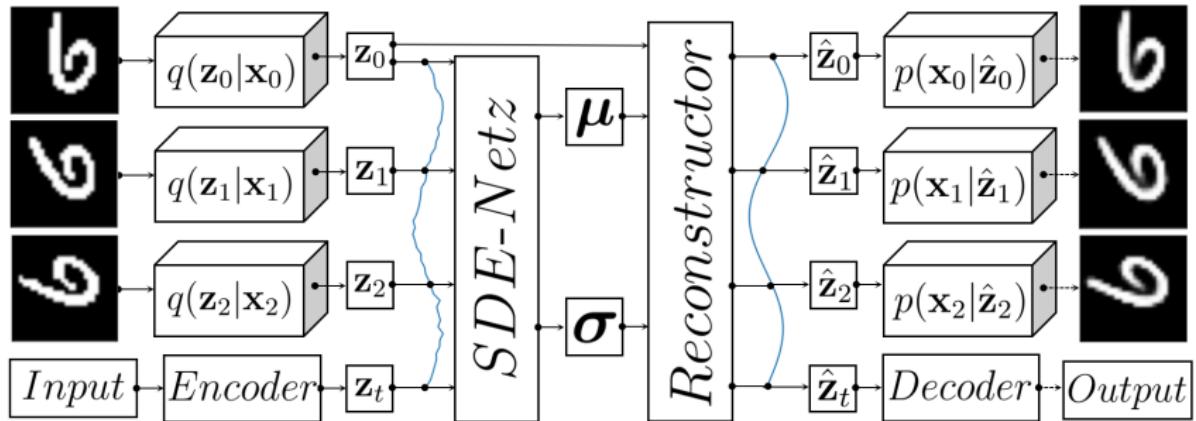
$$X_{m,t_k} := X_{m,t_{k-1}} + \Delta t^m \mu(X_{m,t_{k-1}}) + \sqrt{\Delta t^m} \sum_{j=1}^n \underbrace{\sigma^j(X_{m,t_{k-1}})}_{\sim \mathcal{N}(0,1)} \varepsilon_{j,k}$$

Für $0 = t_0 < \dots < t_m = \tau$ mit $t_k = k \frac{\tau}{m}$ und $\Delta t^m = \frac{\tau}{m}$.

Lernen von $\mu(0)$ und $\sigma(0)$



SDE-VAE Modell für $M = 1$



SDE-Ball-Datensatz

- SDE:

$$dX_t^1 = X_t^2 dt + 0,2 dW_t^1$$

$$dX_t^2 = -X_t^1 dt + 0,1 dW_t^1$$

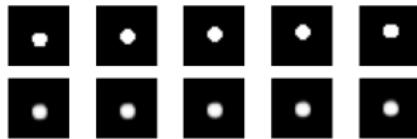
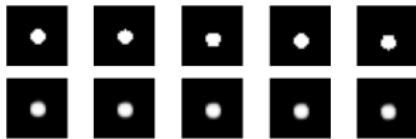
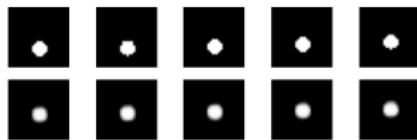
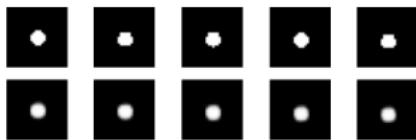
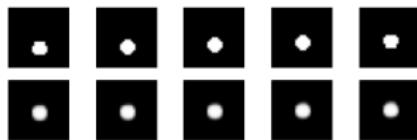
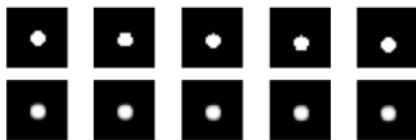
$$X_0 = (0, 1)$$

- Datensatz:

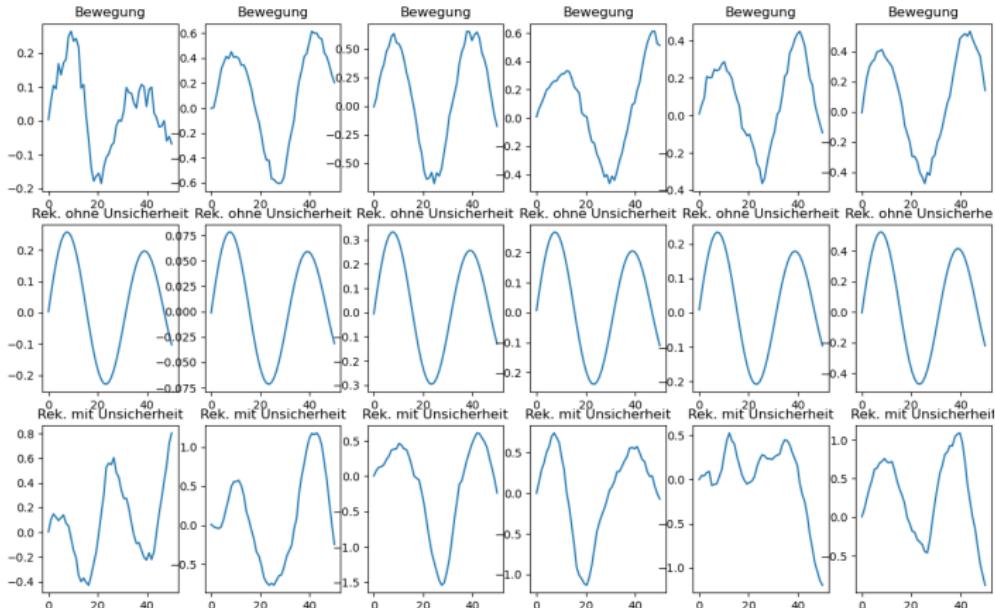
Bilder von einem Ball, dessen Höhe durch X_t^1 gegeben ist.



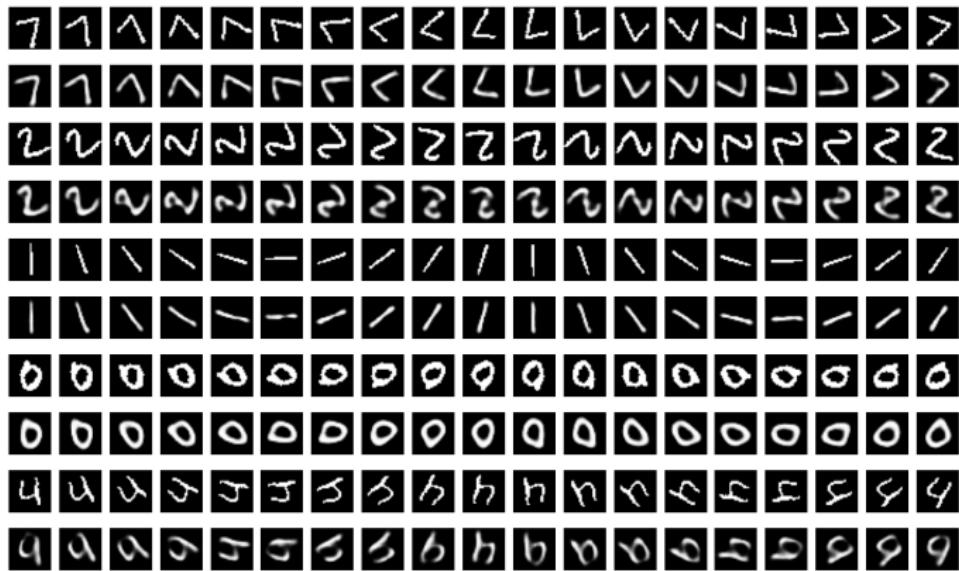
Ergebnisse (SDE-Ball)



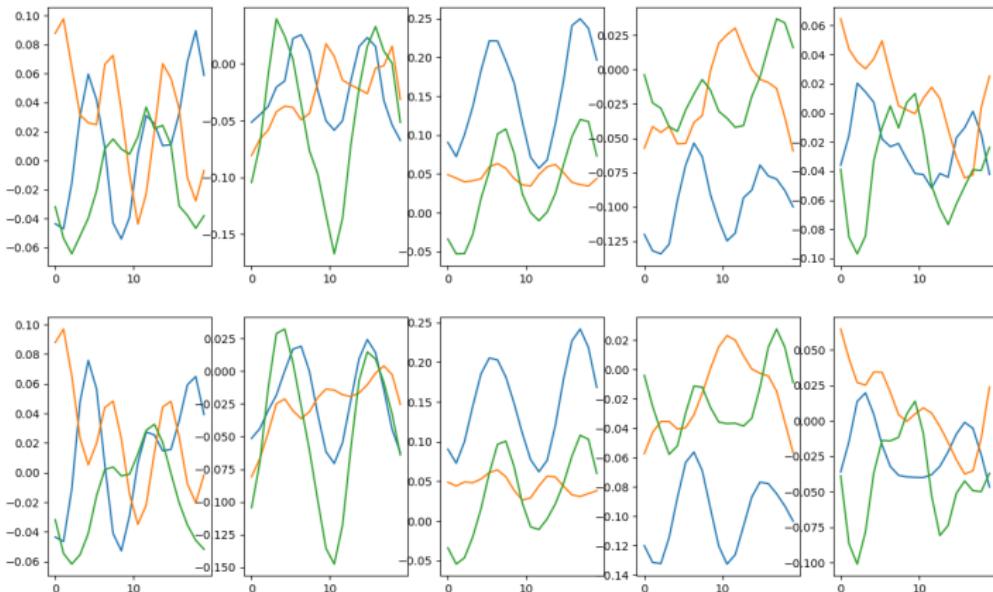
Ergebnisse (SDE-Ball)



Ergebnisse (rotMNIST)



Ergebnisse (rotMNIST)



Quellen und Verweise

- [1] Diederik P. Kingma, Max Welling, *Auto-Encoding Variational Bayes*, 2013, arXiv:1312.6114v10
- [2] Diederik P. Kingma, Max Welling, *An Introduction to Variational Autoencoders*, 2019, arXiv:1906.02691v3
- [3] Çağatay Yıldız, Markus Heinonen, Harri Lähdesmäki, *ODE²-VAE: Deep generative second order ODEs with Bayesian neural networks*, 2019, arXiv:1905.10994v2

Vielen Dank für Ihre Aufmerksamkeit!

