Featureextraktion aus heterogenen multivariaten Zeitreihendaten mechatronischer Systeme mittels Autoencoder-Netzwerken

Autoencoder-based Feature Extraction from Heterogeneous Multivariate Time Series Data of Mechatronic Systems

Karl-Philipp Kortmann, M. Sc.[†‡], Moritz Fehsenfeld, M. Sc.[†] und Dr.-Ing. Mark Wielitzka[†]

[†] Leibniz Universität Hannover, Institut für Mechatronische Systeme, An der Universität 1, 30823 Garbsen, Deutschland
[‡] kortmann@imes.uni-hannover.de

Kurzfassung

Sensor- und Steuerungsdaten moderner mechatronischer Systeme liegen häufig als heterogene Zeitreihen vor und differieren in Abtastrate und Wertebereich. Für prädiktive Aufgaben, etwa im Rahmen eines *Condition Monitorings*, existieren bereits geeignete Klassifikations- und Regressionsverfahren aus dem Bereich des überwachten maschinellen Lernens, deren Leistungsfähigkeit allerdings stark mit der Anzahl gelabelter Trainingsdaten skaliert. Deren Bereitstellung ist oft mit hohem Aufwand in Form von Personenstunden oder Zusatzsensorik verbunden. In dieser Arbeit wird ein Verfahren zur unüberwachten Featureextraktion mittels Autoencoder-Netzwerken vorgestellt, das speziell dem heterogenen Charakter einer Datenbasis Rechnung trägt und die notwendige Menge an gelabelten Trainingsdaten gegenüber bestehenden Verfahren reduziert. Zur Validierung der Ergebnisse werden drei öffentliche Datensätze mechatronischer Systeme aus unterschiedlichen Anwendungsfeldern verwendet.

Abstract

Sensor and control data of modern mechatronic systems are often available as heterogeneous time series with different sampling rates and value ranges. Suitable classification and regression methods from the field of supervised machine learning already exist for predictive tasks, for example in the context of condition monitoring, but their performance scales strongly with the number of labeled training data. Their provision is often associated with high effort in the form of person-hours or additional sensors. In this paper, we present a method for unsupervised feature extraction using autoencoder networks that specifically addresses the heterogeneous nature of the database and reduces the amount of labeled training data required compared to existing methods. Three public datasets of mechatronic systems from different application domains are used to validate the results.

1 Einleitung

Moderne mechatronische Systeme verfügen über eine Vielzahl interner Sensor- und Steuerungsgrößen, auf die etwa über Feldbusschnittstellen zugegriffen werden kann. Zusätzlich lassen sich diese Systeme um externe Messsysteme erweitern, was in Summe zu einem heterogenen Bestand an multivariaten Zeitreihensignalen (*multivariate time series, MTS*) führt, wobei heterogen hierbei divergente Abtastzeiten, Messauflösungen oder Skalenniveaus der einzelnen univariaten Signalen meint.

Im Zuge einer fortschreitenden Digitalisierung werden solche Anlagensignale im zunehmenden Maße für Klassifikationsaufgaben (etwa *Alarm-* oder *Condition Monitoring*) oder zur Schätzung nicht direkt messbarer Zielgrößen (z. B. die Prozessstabilität oder -qualität) eingesetzt. Im Folgenden werden diese zusammenfassend als *Prädiktionsaufgaben* bezeichnet.

Die hierfür primär genutzten Prädiktionsverfahren aus dem Bereich des statistischen beziehungsweise maschinellen Lernens sind gerade bei einer hochdimensionalen, heterogenen Datenbasis auf eine umfangreiche Featureextraktion angewiesen [14]. Dies setzt im Fall einer manuellen Featureextraktion eine hohe fachliche Expertise bezüglich des vorliegenden Systems voraus und geht folglich mit einem hohen Aufwand während der Implementierung einher. Alternativ kommen überwachte End-to-End-Prädiktionsverfahren aus dem Bereich des Deep Learnings zwar explizit ohne zuvor extrahierte Features aus, benötigen aber eine große Anzahl bereits gelabelten Daten für das Training [15].

Methoden des *Representation Learning* haben sich in jüngster Zeit im Bereich der Bild- und Sprachverarbeitung etabliert und ermöglichen eine unüberwachte Featureextraktion, welche die Prädiktionsgüte gängiger Verfahren der manuellen und automatisierten Featureextraktion im Fall von nur wenigen vorhandenen gelabelten Daten übertrifft [1]. Die Anwendung auf mechatronische Systeme beschränkt sich bisher auf spezialisierte Einzellösungen [2], die sich nicht unmittelbar auf beliebige Systeme übertragen lassen.

2 Stand der Forschung

2.1 Representation Learning

Unter dem Begriff des Representation- oder Feature Learnings werden Methoden zusammengefasst, die eine automatische Extraktion relevanter Features ermöglichen und

den Schritt des manuellen Feature Engineerings somit prinzipiell erübrigen. Im Kontext dieser Arbeit liegt der Fokus auf einer unüberwacht arbeitenden Methode, die also ohne Kenntnis der Zielvariable und somit nur anhand der Eingangszeitreihen eine möglichst kompakte Repräsentation der MTS lernt.
Ürsprünglich aus dem Bereich des maschinellen Sehens (*computer vision*) stammend, werden Verfahren des Feature Learnings zunehmend auch für Prädiktionsaufgaben auf Basis von Zeitreihendaten adaptiert; so verwendeten etwa Chen et al. [2] einen Autoencoder zur Featureextraktion aus den Momentensignalen eines 6-Achs-Industrieroboters zur Prädiktion von Kollisionen im Arbeitsraum. Li et al. [18] und Jiang et al. [17] verwenden jeweils ein *Generative Adverssarial Netwerk* (*GAN*) zur Featureextraktion aus industriellen Zeitreihendaten, während Franceschi et al. [1] ein rein Encoder-basiertes Netzwerk in Kombination mit einer sogenannten *Triplet-Loss-Funktion* für die Klassifikation auf diversen Referenzzeitreihen nutzen.
In dieser Arbeit konzentrieren wir uns auf einen Autoencoder-basierten Ansatz, da diese im Allgemeinen im Gegensatz etwa zu GANs eine bessere Repräsentation der Grundgesamtheit der Trainingsdaten ermöglichen, häufig unter Inkaufnahme einer schlechteren Performance als rein generative Modelle [19] (also beim Erzeugen realistischer neuer Zeitreihen), was aber in dieser Arbeit nicht im Fokus steht.

Autoencoder Ein Autoencoder ist ein künstliches Neuronales Netz, dessen primäres Ziel die Rekonstruktion eines Eingangssignales $\mathbf{x}$ darstellt (siehe Abbildung 1). Die dimensionsreduzierte latente Variable (auch *bottelneck* oder *latent space*) $\mathbf{z} = \phi(\mathbf{x}; \theta_{\text{En}})$ ist das Ergebnis einer Encoderfunktion mit den Parametern (Gewichten) θ_{En} und wird im Rahmen des Representation Learnings in dieser Arbeit als Feature herangezogen. Die latente Variable durchläuft während des Trainings anschließend einen Decoder $\tilde{\mathbf{x}} = \psi(\mathbf{z}; \theta_{\text{De}})$ mit den Parametern θ_{De}, der eine Rekonstruktion $\tilde{\mathbf{x}}$ des Eingagnssignales erzeugt. Im vorliegenden Fall von reellen Eingangswerten, wird in der Regel der mittlere quadratische Fehler (*mean squared error, MSE*) als Rekonstruktionsfehler $L_{\text{r}} = \text{MSE}(\mathbf{x}, \tilde{\mathbf{x}})$ verwendet und dient als Verlustfunktion während der Optimierung.

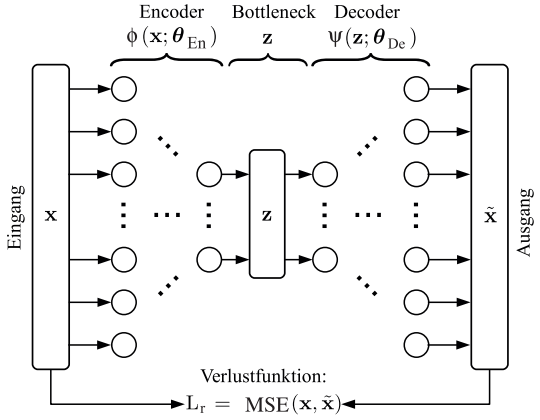


Abbildung 1 Netzwerkdiagramm eines simplen Autoencoders mit eindimensionalem Eingang und angedeuteten FC-Layern in Encoder und Decoder.

Neben vollständig vernetzten Schichten (*fully connected, FC* oder *dense layers*) mit nichtlinearen Aktivierungsfunktionen, finden auch komplexere neuronale Schichten Verwendung. So nutzen Bianchi et al. rekurrente (*RNN*), bidirektionale Schichten in Kombination mit einer zusätzlichen Kernel-Verlustfunktion für die Featureextraktion aus MTS mit fehlenden Werten [7].

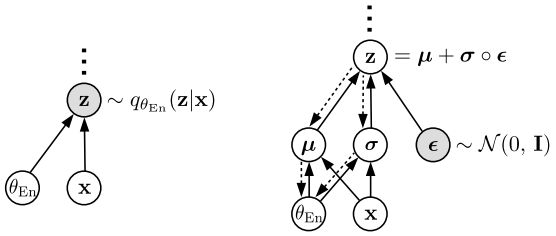
Variational Autoencoder Eine Erweiterung des herkömmlichen Autoencoders stellt der *Variational Autoencoder (VAE)* dar, der hauptsächlich als generatives Modell im Bereich des maschinellen Sehens Anwendung findet. Als Beispiel eines *Variational-Bayes*-Modells modelliert der VAE die unbekannte Verteilungsfunktion der Eingangsdaten $\mathbf{x} \sim p^*(\mathbf{x})$ mittels einer Modellverteilung $p_\theta(\mathbf{x}) \approx p^*(\mathbf{x})$ [12]. Der stochastische Decoder kann somit als bedingte Wahrscheinlichkeitsverteilung $p_{\theta_{\text{De}}}(\mathbf{x}|\mathbf{z})$ aufgefasst werden, die zusammen mit der Prior-Verteilung der latenten Variable $p_\theta(\mathbf{z})$ ein generatives Modell als multivariate Verteilung

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z}) p_{\theta_{\text{De}}}(\mathbf{x}|\mathbf{z}) \tag{1}$$

faktorisiert. Analog dazu stellt der Encoder ein Inferenzmodell dar, das als bedingte Wahrscheinlichkeitsverteilung der latenten Variablen bei gegebenen Eingangsdaten $q_{\theta_{\text{En}}}(\mathbf{z}|\mathbf{x})$ aufgefasst werden kann.
Dieser Ansatz führt über die Anwendung der *evidence lower bound, ELBO* zu der abgewandelten Verlustfunktion [12]

$$L_{\theta_{\text{En}}, \theta_{\text{De}}}(\mathbf{x}) = D_{\text{KL}}(q_{\theta_{\text{En}}}(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) - \mathbb{E}_{q_{\theta_{\text{En}}}(\mathbf{z}|\mathbf{x})}\big(\log p_{\theta_{\text{De}}}(\mathbf{x}|\mathbf{z})\big), \tag{2}$$

mit der Kullback-Leibler-Divergenz D_{KL}, welche die Abweichung zwischen vorgegebener Prior-Verteilung der latenten Variable $\mathbf{z}$ und tatsächlicher Verteilung durch den Encoder bestraft. Der zweite Term repräsentiert den Rekonstruktionsfehler. Als Prior für die latente Variable wird in der Regel die multivariate Standardnormalverteilung $p_\theta(\mathbf{z}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ genutzt.
Für das Training mittels des üblichen stochastischen Gradientabstiegsverfahren (*stochastic gradient descent, SGD*) wird der Gradient der Verlustfunktion $\nabla L_{\theta_{\text{En}}, \theta_{\text{De}}}(\mathbf{x}_{\text{mb}})$ für jedes Mini-Batch an Trainingsdaten $\mathbf{x}_{\text{mb}}$ gebildet, um eine Minimierung der Verlustfunktion in Abhängigkeit der Netzwerkparameter θ_{En} und θ_{De} durchzuführen (*Backpropagation*). Wie Abbildung 2a zu entnehmen ist, ist dies bei einem direkten Ziehen von $\mathbf{z} \sim q_{\theta_{\text{En}}}(\mathbf{z}|\mathbf{x})$ nicht möglich, da die Backpropagation durch die Zufallsvariable $\mathbf{z}$ unterbrochen wird [16]. Erst mit einer mathematisch äquivalente Reparametrisierung durch Auslagerung der Zufallsvariation in die Zufallsvariable ϵ, die in der Regel als normalverteilt modelliert wird, ist eine Rückführung des Fehlers durch den Encoder möglich (sogenannter *reparameterization trick*, siehe Abbildung 2b).



(a) **(b)**

Abbildung 2 **(a)**: Das direkte Sampling von $\mathbf{z}$ macht die Backpropagation (⇢) der Verlustfunktion unmöglich. **(b)**: Erst mit der Reparametrisierung von $\mathbf{z}$ wird die Optimierung der Encoderparameter möglich. Unabhängige Zufallsvariablen sind grau hinterlegt. Basierend auf [13].

3 Methodik

In diesem Abschnitt wird das eigene Autoencoder-basierte Modell, sowie die einbettende Pipeline vorgestellt. Es folgen Erläuterungen zur Versuchsdurchführung, insbesondere zu den gewählten Vergleichsmethoden und den gewählten öffentlich verfügbaren Datensätzen.

3.1 Featureextraktion

Die unüberwachte Extraktion der Features aus einer MTS mit der Anzahl an univariaten Signalen n_{sig} erfolgt mithilfe mehrere VAE, die separat für jede univariate Zeitreihe trainiert werden. Dies ermöglicht im Gegensatz zu gängigen zweidimensional-faltenden Netzarchitekturen (*convolutional neural networks, CNN*) eine Parallelisierung des Trainings und eine individuelle Architektur der Netzwerke, je nach Abtastrate der heterogenen Signale.
Als zusammengefasstes Feature wurden während der späteren Inferenz die 1D-Verkettung der einzelnen latenten Variablen $\mathbf{x}_i,\ i \in \{1, ..., n_{\text{sig}}\}$ genutzt. In Tabelle 1 sind die gewählten (Hyper-)Parameter des entwickelten Modells angegeben. Diese wurden über alle Datensätze konstant gehalten, um eine Allgemeingültigkeit bestmöglich zu demonstrieren.

Anstatt eine konstante Dimension des latenten Raumes vorzugeben, wurde eine konstante Kompressionsrate κ gewählt, sodass die Anzahl an extrahierten Features dynamisch mit der Abtastrate und Signaldauer skaliert. Des Weiteren wurde während Training und Inferenz eine *Instance-Normalization* für jede gefensterte Eingangszeitreihe durchgeführt, um divergenten Wertebereichen der Signale Rechung zu tragen.

Tabelle 1 Zusammenfassung der wichtigsten (Hyper-)Parameter jedes univariaten VAE-Modells

(Hyper-)Parameter	Wert	Bemerkung
Eingangsdimension	$1 \times *$	$*$: maximale Fensterlänge
Kompressionsrate κ	25	Quotient aus $*$ und dim($\mathbf{z}$)
Hidden Layer	$[*/2, */2]$	zwei Hidden Layer
Aktivierungsfunk.	-	tanh bzw. lin. am Ausgang
Regularisierung	-	Early stopping und L_2-Norm
Optimierer	Adam	SGD-Optimierer [20]
Batch-size	$64 - 512$	$\sim n_{\text{train}}$
Normalisierung	-	Instance-/Layer Norm.
Lernrate	1×10^{-4}	kein Lernrate-Scheduler
max. Epochen	1×10^{3}	beachte: Early stopping

3.2 Pipeline

Der Ablauf von Modellierungs-/Trainings- und Inferenzphase ist typisch für ein sogenanntes *semi-supervised* Trainingsverfahren, bestehend aus unüberwachtem Representation Learning und überwachtem Training eines Prädiktors. Nach dem konsekutiven Training beider Modelle werden diese während der Inferenz (hier auf unabhängigen Testdatensätzen) unverändert angewendet. Die Menge an gelabelten Trainingsdaten wird während der Versuchsreihe variiert (logarithmische Skalierung).
Die Feature Learning Methode (VAE) bekommt zuvor sämtliche Trainingsdaten ohne Label zur Verfügung gestellt um Repräsentationen zu lernen. Dies entspricht dem realitätsnahen Anwendungsfall, dass eine Vielzahl an Rohdaten vorliegt, aber nur ein gewisser Anteil davon gelabelt wurde. Für jeden Datensatz, jede Methode und Menge gelabelter Trainingsdaten wurden $n_{\text{repeat}} = 10$ Wiederholungen durchgeführt, sodass empirischer Mittelwert und empirische Standardabweichung der jeweiligen Metrik berechnet werden können. Die Training-/Testaufteilung der Datensätze wurde von den Autoren selbiger übernommen.
Als Entwicklungsumgebung wurde Python 3.8[1] auf einem Rechner mit GPU-Unterstützung (CUDA 7.5) verwendet. Der Quelltext sowie die Links zu den verwendeten Datensätzen werden öffentlich zur Verfügung gestellt[2].

3.3 Vergleichsmethoden

Das vorgestellte Modell sowie die Pipeline werden mit einer Methode der Featureextraktion aus dem aktuellen Stand der Forschung verglichen. Zusätzlich dient die Hauptkomponentenanalyse als deterministische Vergleichsmethode. Die Python-Bibliothek *tsfresh* ermöglicht eine automatische Extraktion und Auswahl statistischer Zeit- und Frequenzbereichsfeatures und stellt somit eine Vergleichsmethode aus dem Bereich der manuellen Featureextraktion dar.
Bei allen Vergleichsmethoden wird ein Ridge-Regression-basierter Prädiktor in derjenigen Parameterisierung verwendet, da sie insbesondere von Autoren mehrerer aktueller Featureextraktoren als favorisierter Prädiktor empfohlen wird. [10, 11]. Für den VAE wird eine *support vector machine (SVM)* mit Gauß-Kern genutzt, um dem erzwungenen Verteilungscharakter der latenten Variable besser Rechnung zu tragen[3].

PCA Mithilfe der Hauptkomponentenanalyse (*principle component analysis, PCA*) lassen sich Zeitreihendaten durch Singulärwertzerlegung in ihrer Dimension reduzieren. Die PCA erzeugt diejenige orthogonale lineare Transformation der Eingangsdaten, welche die Varianz der Kom-

[1] Hierbei insbesondere: torch 1.7.1, sktime 0.4.3 und sklearn 0.24.1.
[2] `https://github.com/MrPr3ntice/vae_rep_learn_mts`
[3] Insbesondere können aufgrund des symmetrischen Kerns im Falle einer Klassifikation geschlossene Intervalle auf einer Variablen separiert werden.

Tabelle 2 Übersicht der ausgewählten Datensätze; $n_{\max}$ gibt die maximale Anzahl an Datenpunkten einer Zeitreihe an.

Datensatz	Trainings-samples	Test-samples	Anzahl Signale	Dauer pro Sample ($n_{\max}$)	Zielgröße (Art)	Referenz
Wälzlagerschäden	1440	800	7 Signale	0.2 s (800)	Wälzlagerzustand (3 Klassen)	[3]
Schrittmotoren	70152	23384	7 Signale	6 ms (60)	Betriebsmodus (4 Klassen)	[4]
Hydraulik	1544	661	17 Signale	60 s (1200)	Kühlerzustand (3 Klassen), Druck im Hydraulikspeicher (Regr.)	[5]

ponenten des Zielunterraums absteigend maximiert. Die somit gewonnene Abbildung kann als Feature für eine Klassifikation oder Regression genutzt werden. Die PCA kann insbesondere als Spezialfall eines linearen Autoencoders ohne Hidden Layer betrachtet werden [12], weshalb sie hier als Vergleichsmethode dienen soll.

Statistische Feature Die Extraktion von manuellen, statistischen Features aus Zeitreihen für ML-Anwendungen ist weit verbeitet. Die Python-Bibliothek *tsfresh* bietet eine Sammlung etablierter Extraktionsmethoden, um eine Vielzahl dieser Features automatisch aus Zeitreihendaten zu generieren und auszuwählen. Die Sammlung der Features enthält z. B. statistische Kennzahlen und Korrelationen im Zeit- und Frequenzbereich. Eine vollständige Übersicht über die Features kann der Dokumentation entnommen werden [8]. Im vorliegenden Fall wurden die Standardeinstellungen[4] genutzt.

ROCKET ***R**and**O**m **C**onvolutional **KE**rnel **T**ransform* ist ein State-of-the-Art-Verfahren, das mithilfe von zufällig gesampleten Faltungskernen Features aus Zeitreihen extrahiert. Anschließend wird eine logistische Regression (Ridge-Regression) mit den erzeugten Features trainiert. Durch diesen vergleichsweise einfachen Aufbau ist der Rechenaufwand von *ROCKET* geringer als bei vergleichbar performanten Verfahren [11]. Zum Stand dieser Arbeit erzeugt diese Methode die durchschnittlich besten Klassifikationsergebnisse auf den Datensätzen des UCR und UEA Zeitreihenarchives [21].

3.4 Datensätze

Die Anzahl der öffentlich zugänglicher Datensätze ist insbesondere im Bereich mechanischer und elektronischer Systemen beschränkt. Zur Validierung der Methode wurden aus den verfügbaren Datensätzen solche ausgewählt, die ein breites Spektrum mechatronischer Anwendungen abgedeckt. Alle Datensätze bestehen aus Messdaten mehrerer Sensoren, die sich z. B. in den Abtastraten unterscheiden. Daraus resultieren drei multivariate Datensätze aus heterogenen Zeitreihen, aus denen drei Klassifikations- und eine Regressionsaufgabe abgeleitet werden. Eine Übersicht der ausgewählten Datensätze findet sich in Tabelle 2.

Wälzlagerschäden Ein weitverbreiteter Anwendungsfall für Verfahren des maschinellen Lernens ist die Erkennung von Wälzlagerschäden. Ein umfassender Referenzdatensatz, der diesen Anwendungsfall abbildet, wurde von Leissmeier et al. [3] veröffentlicht. Zur Schadensklassifikation stehen neben hochfrequent abgetasteten Messungen der Motorströme und Gehäusevibration ($f_s = 64\,\text{kHz}$), zusätzliche, niederfrequentere Daten wie z. B. die Radialkraft ($f_s = 4\,\text{kHz}$) und die Temperatur ($f_s = 1\,\text{Hz}$) zur Verfügung. Der Datensatz umfasst unterschiedliche Schadensarten von Wälzlagern am Außen- und Innenring sowie die zugehörigen Gutdaten unter verschiedenen Betriebsbedingungen. Weitergehende Informationen zum Datensatz sind [3] zu entnehmen.

Schrittmotoren Für die Überwachung von Schrittmotoren existiert ein von Goubeaud et al. [4] publizierter Datensatz. Dieser beinhaltet Messungen des Stroms, der Spannung und der Vibration (Beschleunigung). Die Zielgröße ist der Betriebsmodus des Schrittmotors, der sich zum einen in einen Betrieb im- und gegen den Uhrzeigersinn unterscheidet sowie in einen Betrieb im Normalbereich und jenseits des mechanischen Anschlags. Eine ausführliche Beschreibung des Versuchsaufbaus und der Messdatenaufnahme findet sich in der Erstveröffentlichung [4].

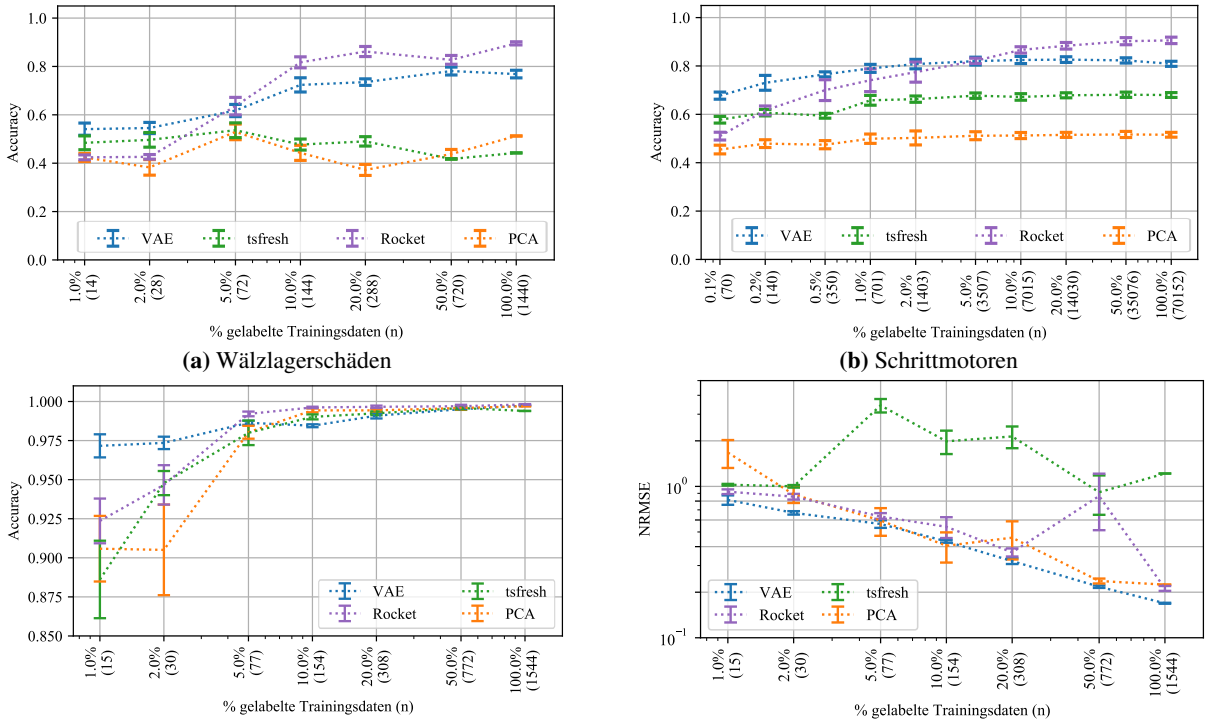
Hydraulik Das in [5] beschriebene hydraulische System ist mit einer Vielzahl verschiedener Sensoren ausgestattet. Neben Messungen des Drucks und der Volumenströme werden zusätzlich Temperatur, Strom und Vibration erfasst. Die Abtastraten reichen von $f_s = 1\,\text{Hz}$ (Temperatur) bis zu $f_s = 20\,\text{Hz}$[5] (Druck), woraus sich ein heterogener Datensatz mit unterschiedlichsten Sensordaten ergibt. Als Zielgrößen sollen an dieser Stelle der Zustand des Kühlers (Klassifikation) und der Druck im Hydraulikspeicher (Regression) betrachtet werden.

4 Ergebnisse

Die Ergebnisse sind in Abbildung 3 (**a**) - (**d**) dargestellt. Dateilierte Ergebnisse können den Tabellen 3 und 4 entnommen werden.
Für die ersten drei Fälle erreicht die Vergleichsmethode ROCKET die höchsten Testergebnisse hinsichtlich der mittleren Accuracy bzw. die niedrigste mittlere Fehlerrate. Lediglich im Fall der Regression des Druckes im Hydraulikspeicher zeigt der VAE fast durchgehend den niedrigsten quadratischen Fehler. Hierzu sei angemerkt, dass ROCKET für Klassifikationsaufgaben entwickelt und bisher mehrheitlich angewendet wurde. Die Hauptkomponentenanalyse und die durch tsfresh ermittelten statistischen Feature sind für die Prädiktionsaufgaben der ersten beiden Datensätze nicht konkurrenzfähig; bei der Klassifikation auf den

[4] Extraktion: `efficient`, Selektion: `extract_relevant_features()`

[5] Hier wurde gegenüber des Originaldatensatzes von 100 Hz auf 20 Hz zugunsten der Rechendauer heruntergesampelt



(a) Wälzlagerschäden

(b) Schrittmotoren

(c) Hydraulik Klassifikation

(d) Hydraulik Regression

Abbildung 3 Ergebnisse der vier verglichenen Featureextraktoren auf den drei Testdatensätzen für variable Anteile an gelabelten Trainingsdaten (logarithmische Skalierung). Die Diagramme **(a) - (c)** zeigen die Accuracy für die Klassifikationsaufgaben; **(d)** zeigt den normalisierten RMSE (log. Skalierung). Dargestellt sind jeweils empirische Mittelwert und Standardabweichung für $n_{\text{repeat}} = 10$.

Hydraulikdaten erreichen alle Methoden für einen höheren Anteil gelabelter Trainingsdaten ähnlich niedrige Fehlerraten.

Bei Betrachtung der Ergebnisse für eine geringe Anzahl gelabelter Trainingsdaten zeigt sich für alle Datensätze, dass die vorgestellte Implementierung des VAEs im Bereich bis mindestens 2 % eine höhere Prädiktionsgüte auf Basis der extrahierten Features bildet. Dies stützt die Forschungshypothese, dass Autoencoder-basierte Methoden des Representation Learnings zur Extraktion von Features aus heterogenen, multivariaten Zeitreihen besonders für sehr kleine Mengen an vorhandenen gelabelten Trainingsdaten geeignet sind.

5 Fazit

In dieser Arbeit wurde ein unüberwachter, Autoencoder-basierter Featureextraktor für Prädiktionsaufgaben auf heterogenen, multivariaten Messzeitreihen mechatronischen Ursprungs vorgestellt und auf drei Datensätzen aus der Forschungsgemeinschaft validiert. Auch wenn die Prädiktionsgüte auf Basis der gesamten Trainigsdaten nicht an aktuelle State-of-the-Art-Extraktoren heranreicht, so konnte doch eine erhöhte Güte für den Fall eines geringen Bestandes an gelabelten Trainingsdaten bei hoher Verfügbarkeit ungelabelter Daten gezeigt werden.

Die hier gezeigten Ergebnisse wurden ausschließlich mit aufbereiteten (Referenz-)Datensätzen erzielt. Ein nächster logischer Schritt hin zu automatisierten Prädiktionsmethoden für technisch-industrielle Anwendungen ist somit die Anpassung der Methode für nicht vorverarbeitete Rohdaten. Hierbei bietet sich etwa das Verfahren des *Denoising Autoencoders* als mögliche Erweiterung an.

6 Literatur

[1] Franceschi, J.-Y.; Dieuleveut, A.; Jaggi, M.: *Unsupervised Scalable Representation Learning for Multivariate Time Series*. Advances in Neural Information Processing Systems. (2019) 32, ISSN 1049-5258.

[2] Chen, T.; Liu, X.; Xia, B.; Wang, W.; Lai, Y.: *Unsupervised Anomaly Detection of Industrial Robots Using Sliding-Window Convolutional Variational Autoencoder*. IEEE Access. (2020) 8, ISSN 2169-3536, S. 47072–47081.

[3] Lessmeier, C.; Kimotho, J.; Zimmer,D.; Sextro, W.: *Condition Monitoring of Bearing Damage in Electromechanical Drive Systems by Using Motor Current Signals of Electric Motors*. European Conference of the Prognostics and Health Management Society. (2016).

[4] Goubeaud, M.; Grunert, T.; Lützenkirchen, J.; Joußen, P.; Ghorban, F.; Kummert, A.: *Introducing a New Benchmarked Dataset for Mechanical Stop Detection of Stepper Motors*. 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS). (2020), S. 1–4.

[5] Helwig, N.; Pignanelli, E.; Schütze, A.: *Condition monitoring of a complex hydraulic system using multivariate statistics*. 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings. (2015) 24, ISSN, S. 210–215.

[6] Dong, Y.; Qin, S. J.: *A novel dynamic PCA algorithm for dynamic data modeling and process monitoring*. Journal of Process Control. (2018) 67, S. 1–11.

[7] Bianchi, F. M.; Livi L.; Mikalsen, K. Ø.; Kampffmeyer M.;

Tabelle 3 Testergebnisse der Klassifikationsaufgaben mit allen drei Datensätzen. Angegeben ist jeweils Mittelwert und Standardabweichung der Accuracy für unterschiedliche Anteile an gelabelten Trainingsdaten.

% gelabelter Trainingssamples (n_{train})	VAE + SVM	PCA + Ridge	tsfresh + Ridge	Rocket + Ridge
Datensatz Wälzlagerschäden				
1.0% (14)	**.541 ± .023**	.424 ± .015	.484 ± .026	.424 ± .009
2.0% (28)	**.545 ± .021**	.383 ± .029	.496 ± .027	.426 ± .008
5.0% (72)	.618 ± .023	.529 ± .029	.536 ± .027	**.636 ± .032**
10.0% (144)	.724 ± .026	.443 ± .028	.477 ± .020	**.817 ± .020**
20.0% (288)	.735 ± .012	.372 ± .020	.490 ± .017	**.862 ± .019**
50.0% (720)	.781 ± .015	.438 ± .017	.417 ± .001	**.827 ± .016**
100.0% (1440)	.768 ± .014	.512 ± .000	.443 ± .000	**.895 ± .006**
Datensatz Schrittmotoren				
0.1% (70)	**.677 ± .013**	.454 ± .016	.578 ± .013	.510 ± .014
0.2% (140)	**.730 ± .025**	.479 ± .014	.607 ± .012	.616 ± .016
0.5% (350)	**.765 ± .010**	.474 ± .015	.594 ± .010	.700 ± .033
1.0% (701)	**.790 ± .015**	.499 ± .017	.658 ± .017	.741 ± .036
2.0% (1403)	**.808 ± .017**	.502 ± .023	.663 ± .012	.774 ± .032
5.0% (3507)	.820 ± .014	.512 ± .015	.677 ± .011	**.821 ± .011**
10.0% (7015)	.825 ± .013	.512 ± .012	.672 ± .013	**.866 ± .012**
20.0% (14030)	.826 ± .011	.515 ± .010	.679 ± .010	**.883 ± .013**
50.0% (35076)	.823 ± .011	.517 ± .012	.681 ± .011	**.902 ± .013**
100.0% (70152)	.809 ± .010	.516 ± .010	.680 ± .010	**.906 ± .012**
Datensatz Hydraulik				
1.0% (15)	**.972 ± .006**	.906 ± .018	.886 ± .021	.924 ± .012
2.0% (30)	**.974 ± .003**	.905 ± .025	.948 ± .007	.947 ± .011
5.0% (77)	.986 ± .001	.980 ± .004	.980 ± .007	**.992 ± .001**
10.0% (154)	.984 ± .001	.994 ± .001	.990 ± .001	**.996 ± .000**
20.0% (308)	.991 ± .002	.994 ± .001	.992 ± .001	**.997 ± .001**
50.0% (772)	.995 ± .000	.996 ± .001	.996 ± .001	**.997 ± .001**
100.0% (1544)	.997 ± .000	.997 ± .000	.994 ± .000	**.998 ± .000**

Tabelle 4 Testergebnisse auf dem Hydraulikdatensatz. Angegeben ist jeweils Mittelwert sowie Standardabweichung des normalisierten RMSE für die Prädiktion des Druckes im Hydraulikspeicher für unterschiedliche Anteile an gelabelten Trainingsdaten.

% gelabelter Trainingssamples (n_{train})	VAE + SVR	PCA + Ridge	tsfresh + Ridge	Rocket + Ridge
1.0% (15)	**.814 ± .058**	1.674 ± .350	1.026 ± .013	.923 ± .033
2.0% (30)	**.669 ± .018**	.885 ± .107	1.002 ± .019	.856 ± .037
5.0% (77)	**.565 ± .033**	.595 ± .123	3.429 ± .350	.633 ± .022
10.0% (154)	.433 ± .009	**.405 ± .091**	1.986 ± .350	.541 ± .085
20.0% (308)	**.321 ± .014**	.458 ± .129	2.138 ± .350	.367 ± .023
50.0% (772)	**.217 ± .004**	.237 ± .009	.915 ± .266	.864 ± .350
100.0% (1544)	**.169 ± .001**	.224 ± .000	1.217 ± .000	.212 ± .008

Jenssen, R.: *Learning representations of multivariate time series with missing data*. Pattern Recognition. (2019) 96, ISSN 0031-3203.

[8] Christ, M.; Braun, N.; Neuffer, J.; Kempa-Liehr A.W.: *Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package)*. Neurocomputing. (2018) 307, ISSN 1573-756X, S. 72–77.

[9] Dau, H. A. and Bagnall, A. and Kamgar, K. and Yeh, C. M. and Zhu, Y. and Gharghabi, S. and Ratanamahatana, C. A. and Keogh, E.: *The UCR time series archive*. IEEE/CAA Journal of Automatica Sinica. (2019) 6, S. 1293–1305.

[10] Le Nguyen, T.; Gsponer, S.; Ilie, I.; O'Reilly, M.; Ifrim, G.: *Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations*. Data Mining and Knowledge Discovery. (2019) 33, ISSN 1573-756X, S. 1183–1222.

[11] Dempster, A.; Petitjean, F.; Webb, G. I.: *ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels*. Data Mining and Knowledge Discovery. (2020) 34, ISSN 1573-756X, S. 1454–1495.

[12] Kingma, D. P.; Welling, M.: *An Introduction to Variational Autoencoders* . Foundations and Trends in Machine Learning. (2019) 12, No. 4, S. 307–392.

[13] Kingma, D. P.; Welling, M.: *Auto-encoding Variational Bayes*. (2013), arXiv preprint arXiv:1312.6114.

[14] Fawaz, H. I.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P. A.: *Deep learning for time series classification: a review*. Data Mining and Knowledge Discovery. (2019), 33(4), S. 917–963.

[15] Lei, Y.; Jia, F.; Lin, J.; Xing, S.; Ding, S. X.: *An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data*. IEEE Transactions on Industrial Electronics. (2016), 63(5), S. 3137–3147.

[16] Rezende, D. J.; Mohamed, S., Wierstra, D.: *Stochastic back- propagation and approximate inference in deep generative models*. In: International Conference on Machine Learning. (2014), S. 1278—1286.

[17] Jiang, W.; Cheng, C.; Zhou, B.; Ma, G.; Yuan, Y.: *A novel GAN-based fault diagnosis approach for imbalanced industrial time series*. (2019), arXiv preprint arXiv:1904.00575.

[18] Li, D.; Chen, D.; Jin, B.; Shi, L.; Goh, J.; Ng, S. K.: *MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks*. In: International Conference on Artificial Neural Networks. (2019), Springer, Cham, S. 703–716.

[19] Grover, A.; Dhar, M.; Ermon, S.: *Flow-GAN: Combining maximum likelihood and adversarial learning in generative models*. In: Proceedings of the AAAI Conference on Artificial Intelligence. (2018) 32, 1.

[20] Kingma, D. P.; Ba, J.: *Adam: A method for stochastic optimization*. (2014), arXiv preprint arXiv:1412.6980.

[21] Ruiz, A. P.; Flynn, M.; Large, J.; Middlehurst, M.; Bagnall, A.: *The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances*. Data Mining and Knowledge Discovery. (2020), Springer OA, ISSN: 1573-756X, S. 1–49.