Hochschule
Albstadt-Sigmaringen

Albstadt-Sigmaringen University

Fakultät - Engineering

Studiengang: Data Engineering & Consulting M.Sc.

Semesterbegleitende Projektaufgabe

Machine Learning in Manufacturing Hausarbeit

Eingereicht

im Sommersemester 2023

am 04.11.2023

Prüfer: Felix Georg Müller (Plus10 GmbH)

Vorname	**Nachname**	**Matrikelnummer**
Hans-Joachim	Redeker	74464
Johannes	Lehner	103854
Simon	Köller	103605

Inhaltsverzeichnis

Themenzuordnung

Bearbeiter	**Themen:**
Simon Köller	Data Extraction, Data Pre-processing
Hans-Joachim Redeker	Data-Preprocessing, Data-Visualisierung
Johannes Lehner	LSTM-Modell, Evaluierung

Abbildungsverzeichnis

1 Einführung (Aufgabenstellung)

Der Ausfall von einer oder mehreren Maschinen kann Unternehmen schnell einige tausend Euro kosten (adtance.com, 2020). Daher ist die Vermeidung von Stillständen von großer Bedeutung. Moderne Technologien ermöglichen das Analysieren von Maschinendaten, um frühzeitig Wartungsarbeiten an Maschinen bzw. ihren Bauteilen durchführen zu können. Dieses Verfahren wird als Predictive Maintenance bezeichnet (compamind.de, 2023).

Die nachfolgende Arbeit beinhaltet die Dokumentation der semesterbegleitenden Aufgabe der Lehrveranstaltung „Machine Learning in Manufacturing“. Das Szenario der Aufgabenstellung ist, dass belastete Kugellager in Elektromotoren als Ursache für den Ausfall einer Produktionslinie identifiziert werden konnten. Die Aufgabe besteht darin, ein Vorhersagemodell zu entwickeln, welches die Remaining Useful Lifetime (RUL), also die Restnutzungsdauer, der Kugellager (Bearings) vorhersagt.

Die Daten für die vorliegende Arbeit wurden mit Hilfe der Testplattform „Pronostia“ generiert, welche in der nachfolgenden Abbildung zu sehen ist. Die Plattform ist eine Entwicklung der Abteilung „Automatic control and Micro-Mechatronic Systems“ des FEMTO-ST Instituts aus Franche-Comté. Ein rotierender Part verursacht die Lagerrotation. Des Weiteren ist ein verschleißerzeugender Part an der Plattform angebracht. Hierbei übt ein pneumatischer Zylinder eine Radialkraft auf die Außenringe der Lager aus. Zur Generierung der Daten sind Beschleunigungssensoren angebracht. Das Vibrationssignal wird mit einer Frequenz von 25,6 kHz erfasst. Das Aufnahmeintervall hat eine länge von 10s, wobei die Aufnahmelänge nur 0,1s beträgt. Pro Aufnahmezyklus werden daher 2560 Geschwindigkeitssignale aufgenommen. Die Messung wurde so lange durchgeführt, bis die Amplitude der Vibration Werte über 20g angenommen hat. Nicht jedes Kugellager wurde bis zum Ausfall gemessen. Insgesamt stehen für diese Aufgabe von 17 verschiedenen Kugellagern Messdaten zur Verfügung. Dabei wurden die Kugellager mit drei unterschiedlich Kräften beansprucht.

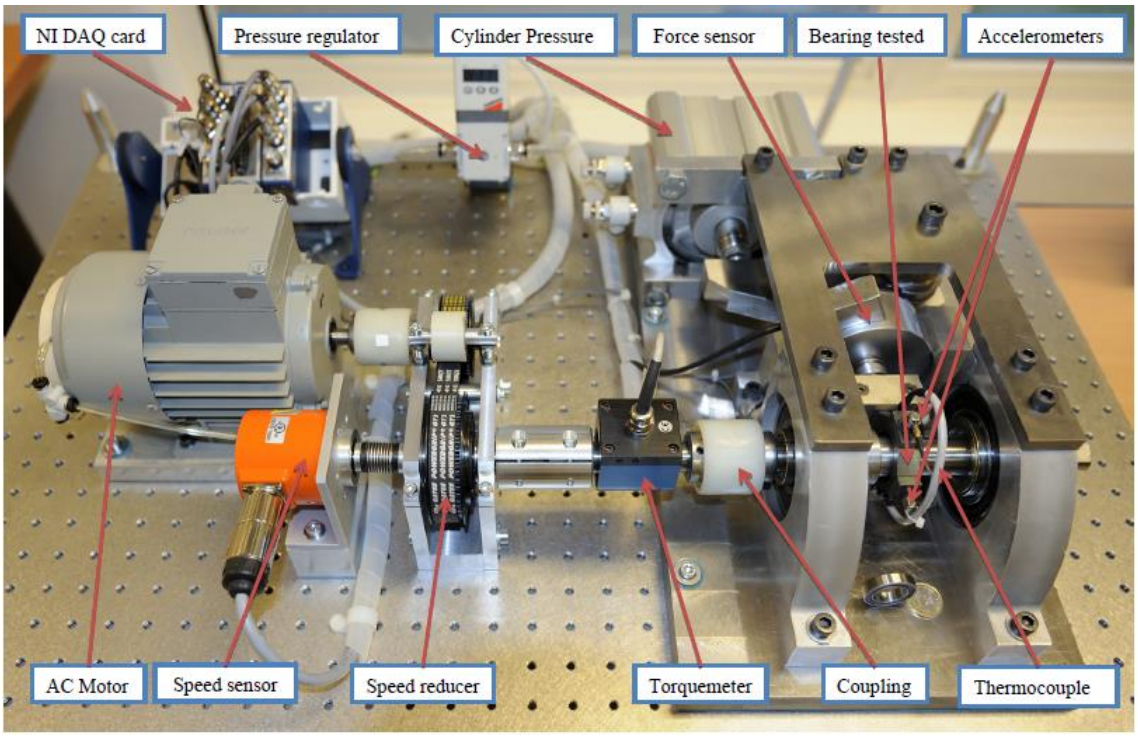

Abbildung 1: Pronostia-Testplattform (IEEE Reliability Society, FEMTO-ST Institute, 2012)

Die für die Aufgabe notwendigen Daten wurden von einem GitHub Repository bezogen (https://github.com/wkzs111/phm-ieee-2012-data-challenge-dataset). Für jeden Messzyklus steht eine csv-Datei zur Verfügung. Jedes Kugellager hat seinen eigenen Unterordner, in dem alle dazugehörigen csv-Dateien abgespeichert sind. Übergeordnet sind die Daten in Learning und Test-Set in entsprechend zwei Ordnern geteilt.

Um mit den Daten in Python arbeiten zu können, sind für jedes Kugellager alle csv-Dateien als pandas DataFrame eingelesen und zusammengefügt worden. Anschließend wurden die 17 DataFrames in pickle-Files exportiert, welche die Ladezeit verkürzen.

2 Data Understanding und Preprocessing

In Abbildung 1 ist ein Überblick über die Spalten des Rohdatensatzes dargestellt. Bereits beim Einlesen der Datensätze wird eine Spalte „bearingName“ eingefügt, um sie gegebenenfalls beim Zusammenfügen der Datensätze verwenden zu können.

	bearingName	hour	minute	second	microsecond	horizontalAcceleration	verticalAcceleration
0	Bearing1_1	9	39	39	65664.0	0.552	-0.146
1	Bearing1_1	9	39	39	65703.0	0.501	-0.480
2	Bearing1_1	9	39	39	65742.0	0.138	0.435
3	Bearing1_1	9	39	39	65781.0	-0.423	0.240
4	Bearing1_1	9	39	39	65820.0	-0.802	0.020

Abbildung 2: Übersicht über Spalten des Rohdatensatzes

Die Zeitangaben liegen in einzelnen Spalten „hour“, „minute“, „second“ und „microsecond“ vor. Bis auf „microsecond“ welche als float-Dateityp vorliegt, liegen die anderen drei als int64-Dateityp vor. Diese werden mit Hilfe einer Funktion zu einem „timestamp“ zusammengefasst und als gleichnamige Spalte dem Datensatz hinzugefügt. Das Ergebnis liegt als int64-Dateityp vor und bezieht sich auf den 02.01.1970 als Startwert. Die hohe Zahl sind die Mikrosekunden von diesem Datum bis zu dem Zeitpunkt, an dem der Messwert aufgenommen wurde.

Die Messerwerte der Beschleunigungssensoren werden als „horizontalAcceleration“ und „verticalAcceleration“ aufgeführt. Um den Datensatz zu vereinfachen werden die resultierende Kräfte über den Satz des Pythagoras ausgerechnet und als „totalAcceleration“ im Datensatz erfasst. Um Ausreißer zu entfernen, wird eine MovingAverage-Methode auf die „totalAcceleration“-Spalte angewendet und die Werte als neue Spalte in den Datensatzeingefügt. Die MovingAverage-Methode berechnet aus einem definierten Bereich um einen Wert herum den Mittelwert und trägt diesen in der Spalte ein. Anschließend springt er einen Wert weiter und berechnet erneut den Mittelwert. Dieser Vorgang iteriert durch die gesamte Spalte hindurch bis alle Werte berechnet sind, so bleibt die Anzahl an Werten erhalten allerdings findet durch die Mittelwertsberechnung eine Glättung des Datensatzes statt. Die zunächst konstanten Werte in der Spalte „totalAcceleration-MovingAverag“ kommen durch die Verwendung einer Fill-Methode Zustande, da der MovingAverage zunächst NaN-Werte generiert, bis er die volle Breite des zu berechnenden Bereichs erreicht hat.

Um weitere Feature zu erhalten und für eine eventuelle Verwendung von Cassifier-Algorithemen wird ein „StandartScaler“ auf die Spalte „totalAccelerationMovingAverage“ angewendet. Die Spalte „Defect“ bezieht sich ebenfalls auf die „totalAccelerationMovingAverage“ und hat für alle Werte unter 20G eine 0 und für alle über 20G eine 1. Mit Hilfe dieser Spalte kann über eine Funktion die RUL berechnet werden. In einem weiteren Schritt werden die RUL in drei unterschiedliche Bereiche aufgeteilt. Von 100% bis 40% der RUL ist das Kugellager „uncritical“ im Bereich 40% bis 20% ist 'wear recognizable' und ab 20% „failure soon“. Die letzte Spalte „ID“ dient zur Komprimierung der Datensätze in der Feature Extraction, da die Zeilen mit gleichem ID-Wert zu einem Wert im extrahierten Feature-Datensatz zusammengefasst werden.

	timestamp	totalAcceleration	totalAccelerationMovingAverage	totalAcceleration_stan	Defect	RUL	Status	ID
0	34779065664	0.570982	0.633356	-0.100987	0	27650062966	uncritical	1
1	34779065703	0.693831	0.633356	-0.100987	0	27650062927	uncritical	1
2	34779065742	0.456365	0.633356	-0.100987	0	27650062888	uncritical	1
3	34779065781	0.486342	0.633356	-0.100987	0	27650062849	uncritical	1
4	34779065820	0.802249	0.633356	-0.100987	0	27650062810	uncritical	1

Abbildung 3: Übersicht über der ersten 5 Zeilen nach der Datapreparation

2.1 Preprocessing für das LSTM-Modell

Um abschließend die Daten explizit für das LSTM-Modell anzupassen, mussten noch weitere Schritte durchgeführt werden. Da bei der Feature-Extraction auch die RUL auf mehrere Features extrahiert wurde, mussten diese entfernt werden damit keine RUL-Daten in den X-Werten gespeichert sind. Es wurden alle Features die RUL im Namen haben entfernt, außer die für die Y-Wert benötigte „RUL__mean“. Anschließend wurden je nach Versuch aus verschiedenen Bearings-Datensätzen, ein Training und ein Test-Datensatz gebildet. Anschließend wurden diese in x_train, x_test, y_train und y_test übergeben und standardisiert skaliert zwischen -1 und 1.

Im nächsten Schritt wurden aus den 2-dimensionalen Datensätzen 3-dimensionale Numpy Arrays gebildet, da diese für die Sequenzangabe bei einem LSTM benötigt wird. Da durch die Sequenz wenige letzte Reihen der x-Daten verloren gehen können, wurde die die Reihenanzahl bei dem y-Werten übergeben.

Aufgrund der großen Anzahl an Datenreihen wurden diese gedroppt, mit der Annahme, dass aufgrund der großen Anzahl an Daten und den am Ende stehenden negativen RULs, keine große Gewichtung hat.

3 LSTM-Modell (Long Short-Term Memory) (Johannes Lehner)

Das LSTM-Modell ist auf einen Recurrent Neural Network aufgebaut und wird besonders dann genutzt, wenn ein Muster in Datensequenzen zu erkennen ist. (Databasecamp.de, 2022) Das Modell eignet sich sehr gut da die timestamp bzw RUL eine zeitliche Sequenz darstellt und somit immer eine Vorherige Sequenz miteinbezogen wird. Als Code-Grundlage, wie ein LSTM für die RUL-Vorhersage verwendet werden kann, wurde ein Bsp. von Github von dem User „Archd3sai" verwendet (archd3sai, 2020). Allerdings wurden einige Veränderungen durchgeführt, um es an diesen Fall anzupassen. Die Modellarchitektur ist wie folgt aufgebaut.

1. **LSTM-Layer** mit **128 Units, return_sequences = True**, und dem **input_shape = (sequence_length, nb_features):** Dabei entspricht die sequence_length die Anzahl an Sequenzen, welche bei der Berechnung rückläufig betrachtet werden. Diese ist in diesem Fall standardmäßig auf 50 gesetzt. Nb_features entspricht der Anzahl der Features, die in x Trainingsdaten vorhanden sind. Dies entspricht genau 39 in allen Versuchen. Die LSTM-Schicht erfasst Abhängigkeiten zwischen den Zeitschritten in dem es die letzten 50 Zeilen für die aktuelle RUL-Prediction berücksichtig. LSTM-Layer haben standardmäßig die Aktivierungsfunktion „tanh" welche beibehalten wird. Für jedes Layer wird eine Dropoutrate festgelegt, welche sich zu den unterschiedlichen Versuchen verändert.

2. **LSTM-Layer** mit **128 Units** und **return_sequences = False**: Diese Schicht erweitert die Möglichkeit, langzeitige Abhängigkeiten zu speichern. Dadurch können komplexere zeitliche Abhängigkeiten erkannt werden.

3. **Dense-Layer** mit **einer Unit** und einer **linearen Aktivierungsfunktion:** Dieses Layer soll die RUL schlussendlich berechnen, und da pro Zeile nur ein Wert ausgegeben wird, und deshalb wird nur eine Unit angesetzt.

4. **Dense-Layer** mit **128 Units** und der **tanh Aktivierungsfunktion:** Dieser Dense-Layer identifiziert wichtige Merkmale aus den Features aus X_Train Daten und hebt diese hervor.

5. **Compile:** Am Ende wird das Modell kompiliert. Dabei wird MSE als Verlustfunktion verwendet, der Algorithmus wird mit RMSprop verwendet, um den Verlust zu vermindern und es wird bei den metrics der MSE eingetragen, welcher Wert berechnet wird, um das Modell zu überwachen.

Im nächsten Schritt wird der Befehl gegeben das Training durchzuführen. Dabei werden die Trainingsdaten x_train und y_train übergeben. Des Weiteren werden die Anzahl der Epochen, die batch_size von 32 und ein validation_split von 0,1 mitgegeben. Außerdem wird ein callback festgelegt, der sagt dass wenn der Validierungs-Verlust in einer Epoche nichtmehr verbessert wird, wird das Training abgebrochen.

3.1 Definitionen der KPI`s

Alle KPI’s werden mit den skalierten Werten zwischen -1 und 1 berechnet. Die KPI‘s müssen für die Evaluierung immer im Zusammenhang mit der Skalierung betrachtet werden.

Mean Squared Error (MSE): Der Mean Squared Error berechnet den Mittelwert des quadrierten Unterschieds zwischen der Prediction und den tatsächlichen Werten (Valerie Benning, 2020).

Root Mean Squared Error (RMSE): Der RMSE ist die Quadratwurzel des Durchschnitts der quadrierten Fehler. Der Unterschied zum MSE ist, dass deutliche Fehler mehr Gewichtung haben (Nicolas Vandeput, 2020).

Mean Absolute Difference (MAE): Dies KPI gibt den Mittelwert der absoluten Fehler an (Nicolas Vandeput, 2020).

Mean Absolute Percentage Error (MAPE): Wird für die Prognosegenauigkeit verwendet. Dabei wird die Summe der absoluten Fehler durch die Summe der tatsächlichen Werte geteilt. Dieser gibt den durchschnittlichen Prozentualen Fehler der Vorhersage wieder (Nicolas Vandeput, 2020).

4 Evaluation (Johannes Lehner)

Aufgrund von unterschiedlicher Formatierung bei der Timestamp der Rohdaten wurden die folgenden Kugellager in den Versuchen nicht berrücksichtigt:

Bearing_1_7, Bearing_2_4, Bearing_3_2

4.1 Erster Versuch: Condition 1 und 2 kombiniert

Im ersten Versuch wurden alle Bearings aus der ersten und zweiten Condition verwendet, um die RUL vorherzusagen:

Trainings-Daten: Bearing_1_1, Bearing_1_2, Bearing_2_6, Bearing_2_5, Bearing_2_3, Bearing_1_6, Bearing_1_5, Bearing_1_4, Bearing_2_2, Bearing_2_1

Test-Daten: Bearing_1_3, Bearing_2_7

Des Weiteren wurden folgende Parameter verwendet:

- **Sequenzlänge:** 50
- **Dropout (für alle Layer):** 0,2
- **Epochen:** 5

Ergebnisse:

Folgende Abbildung zeigt den Vergleich der geschätzten RUL (in rot) des Modells und der tatsächlichen RUL (in blau) der Test-Bearings. In dem Schaubild ist zu erkennen, dass die vorhergesagte RUL im letzten Drittel nah an der tatsächlichen RUL liegt. In den ersten zwei Dritteln ist die geschätzte RUL stark abgewichen von der richtigen RUL. Die tatsächliche RUL verläuft nicht ganz linear, da zwei verschiede Bearings-Daten gleichzeitig im Test-Datensatz verwendet wurden.

Abbildung 4: RUL-Prediction V.1 Bearing_1_3, Bearing_2_7

Die Verzerrung der vorhergesagten RUL zeigen, dass das Modell leicht over fitted ist. Angenommen wird, dass es an der Menge der Input-Daten liegt, die für das Training angesetzt wurden.

Zur Evaluierung wurden noch weitere Kennzahlen berechnet:

- MSE: 0.4929613
- RMSE: 0.7031042
- MAE: 0.5635462
- MAPE: 531.39812%
- RUL-Prediction: 2:5:1:340000

Im Verhältnis zu der Skalierung schneiden die KPI's relativ schlecht hab. Insgesamt kann dieser erste Versuch aufzeigen, dass das Modell eine RUL annäherungsweise bei dem ersten Drittel zutrifft. In Versuchen danach wurden die Epochen und die Dropoutrate deutlich erhöht, allerdings hat dies dazu geführt, dass das Training weitaus drastischer over fitted wurde. Deshalb werden diese Versuche nicht weiter dokumentiert.

4.2 Versuch 2: Condition 1

Mit der Annahme, dass zu viele Daten in Versuch 1 vorhanden waren und es ohnehin Unterschiede aufgrund der Condition gibt, wurden in diesem Versuch alle Kugellager der Condition 1 untersucht. Des Weiteren wurde nur ein Bearing-Datensatz als Test-Datensatz übergeben, da dies zu Ungenauigkeit der tatsächlichen RUL geführt hat.

Trainings-Daten: Bearing_1_1, Bearing_1_2, Bearing_1_4, Bearing_1_5, Bearing_1_6

Test-Daten: Bearing_1_3

Des Weiteren wurden folgende Parameter verwendet:

- **Sequenzlänge:** 50
- **Dropout (für alle Layer):** 0,2
- **Epochen:** 5

Ergebnisse:

Wie in der Abbildung zu erkennen ist die geschätzte RUL (rote Linie) deutlich näher an der tatsächlichen RUL (blaue Linie). Was zu erkennen ist das umso größer die RUL ist umso ungenauer werden die Prediction, und bei den ersten Units sieht es aus, als würde Over Fitting bestehen bzw. wird es deutlich ungenauer.

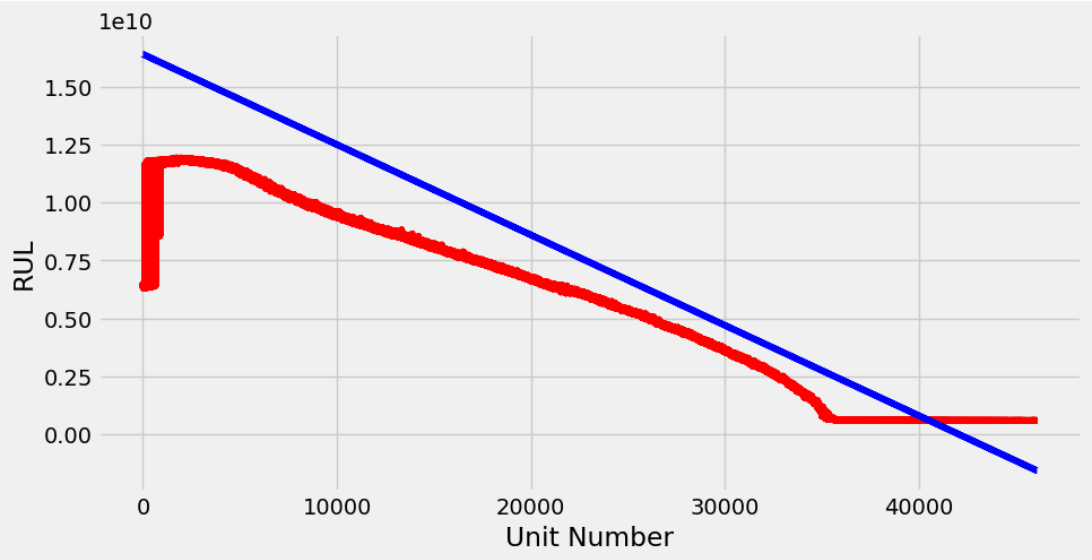

Abbildung 5: RUL-Prediction V.2 Bearing_1_3

Zur Evaluierung wurden noch weitere Kennzahlen berechnet:

- MSE: 0.0659257
- RMSE: 0.2567600
- MAE: 0.2216560
- MAPE: 156.03706%
- RUL-Prediction: 2:46:12:140000– 4:15:14:320000

Auffällig ist, dass sich MSE und RMSE stark unterscheiden. Dadurch wird nochmal bestätigt, dass die größeren Fehler bei steigender RUL schlechter werden, und durch diese Ungleichmäßigkeit führt zu dem Unterschied von MSE und RMSE. Aufgrund dieses Ergebnisses wurde die Annahme getroffen, dass zuerst die niedrigen Werte der RUL im Modell trainiert werden und dass durch mehr Epochen ein besseres Ergebnis realisiert werden kann.

4.3 Versuch 3: Condititon 1 und 7 Epochen

Mit der Annahme, dass mehr Epochen ein besseres Ergebnis liefern, wurde Versuch 3 durchgeführt. Alles andere wurde beibehalten

Trainings-Daten: Bearing_1_1, Bearing_1_2, Bearing_1_4, Bearing_1_5, Bearing_1_6

Test-Daten: Bearing_1_3

Des Weiteren wurden folgende Parameter verwendet:

- **Sequenzlänge:** 50
- **Dropout (für alle Layer):** 0,2
- **Epochen:** 7

Ergebnisse:

Die Annahme, dass mehr Epochen zu einem besseren Ergebnis führen, hat sich bestätigt. Wie im Schaubild zu erkennen ist hat sich die Vorhersage (rote Linie) vor allem bei den Startwerten, mit der hohen RUL verbessert, womit die Ähnlich-

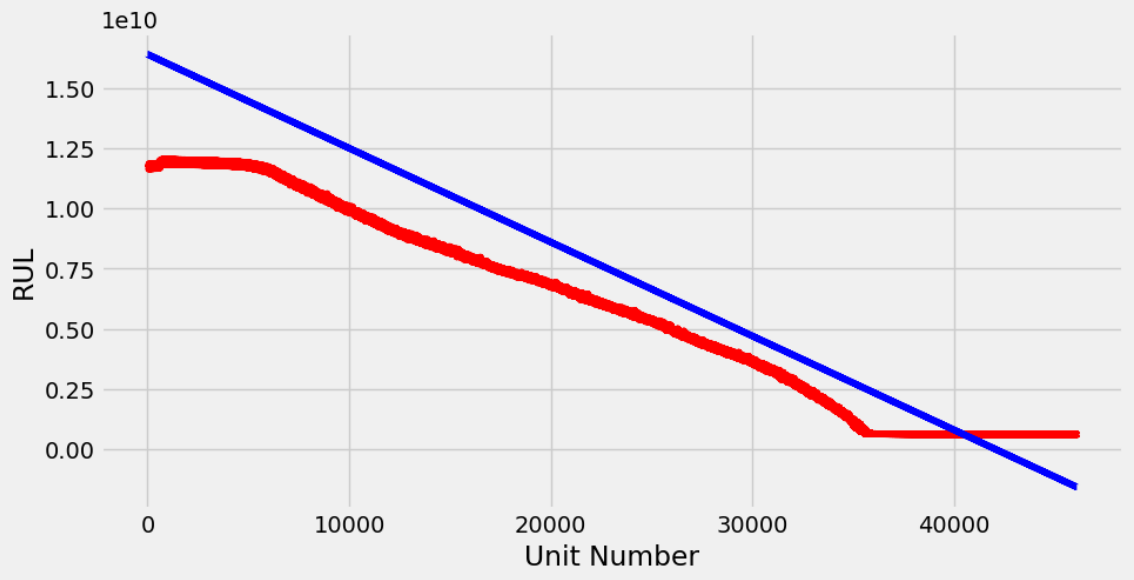

Abbildung 6: RUL-Prediction V.3 Bearing_1_3

keit der Vorhersage deutlich zu erkennen ist.

Zur Evaluierung wurden folgende KPI`s berechnet:

- MSE: 0.0520238
- RMSE: 0.2280974
- MAE: 0.2054249
- MAPE: 157.63036%
- RUL-Prediction: 4:16:13:640000

Die Kennzahlen verdeutlichen durch eine kleine Verbesserung, dennoch bedeutsame Verbesserung des Modells. Nur der MAPE wurde um ca. 1 Prozentpunkt schlechter, was jedoch im Vergleich zu den anderen Verbesserten weniger an Gewichtung verliert. Dieses Ergebnis stellt das beste Ergebnis für Versuche mit Condition 1 dar. Es wurde noch ein Versuch mit 10 Epochen durchgeführt,

welcher aber zu Over Fitting und somit zu schlechteren Ergebnissen geführt hat. Dieser wird aufgrund dessen nicht aufgezeigt.

4.4 Versuch 4: Condition 2

Aufgrund der Tasache, dass Condition 1 gute Ergebnisse liefern konnte, wurde das gleiche auf Condition 2 angwendet. Und der Versuch wurde wie folgt aufgebaut.

Trainings-Daten: Bearing_2_6, Bearing_2_5, Bearing_2_2, Bearing_2_1

Test-Daten: Bearing_2_3

Des Weiteren wurden folgende Parameter verwendet:

- **Sequenzlänge:** 50
- **Dropout (für alle Layer):** 0,2
- **Epochen:** 5

Ergebnisse:

Wie in der Abbildung zu erkennen ist, ist die zweite Hälfte der Datenreihen sehr nahe an den Ergebnissen, wenn man die Vorhersagewerte (rot) und die tatsächlichen Werte(blau) vergleicht, und die erste Hälfte mit dem größeren RUL-Werten stark schwankend.

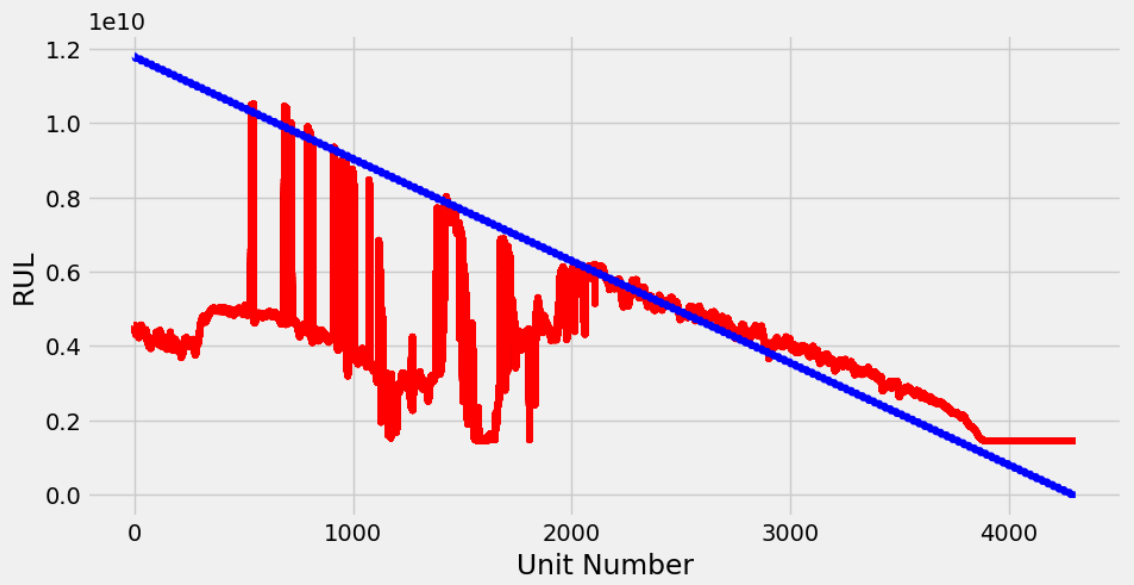

Abbildung 7: RUL-Prediction V.4 Bearing_2_3

KPI’s zur Evaluierung:

- MSE: 0.3432625
- RMSE: 0.5858860
- MAE: 0.4089897
- MAPE: 106.95454%
- RUL-Prediction: 2:10:39:960000

Zu erkennen sind hohe MSE, RMSE und MAE-Werte im Vergleich zu der MAPE. Aus diesen und ähnlichen Versuchen mit mehr Epochen, unterschiedlichen Drop-out-werten, sowie Sequenzlängenveränderung haben sich immer ähnliche Ergebnisse gezeigt und die meisten davon waren schlechter. Dann wurde erkannt, dass die RUL-Kugellager von Condition 2 und aber auch 3 stark schwanken. Bspw. hat Bearing_2_5 eine RUL von 2e10 und Bearing 2_7 eine RUL von 0,19e10. Diese Starken Schwankungen haben immer wieder für ungenaue Predictions für immer größer werdende RUL gesorgt.

4.5 Versuch 5: Condition 2 – weniger Kugellager

In diesem Versuch wurden aufgrund der Tatsache, dass mit allen Kugellagern der Condition 2 immer zu schlechten Ergebnissen kam, wurde welche mit stark abweichender tatsächlichen RUL nicht weiterverwendet. Und damit die vorhergesagte RUL sich mehr an den Werten davor orientiert wurde eine Sequenzlänge von 100 festgelegt. Aufgrund des daraus folgenden kleineres Trainingsdatensatz wurde die Epochenanzahl auf 3 verringert.

Trainings-Daten: Bearing_2_1, Bearing_2_3

Test-Daten: Bearing_2_2

Des Weiteren wurden folgende Parameter verwendet:

- **Sequenzlänge:** 100
- **Dropout (für alle Layer):** 0,2

- **Epochen:** 3

Ergebnisse

Die Abbildung zeigt, dass die Steigung ähnlich der berechneten RUL (rot) der richtigen RUL (blau) ähnlich ist. Jedoch ist die geschätzte RUL stetig um ca 0.2e10 Zeiteinheiten geringer.

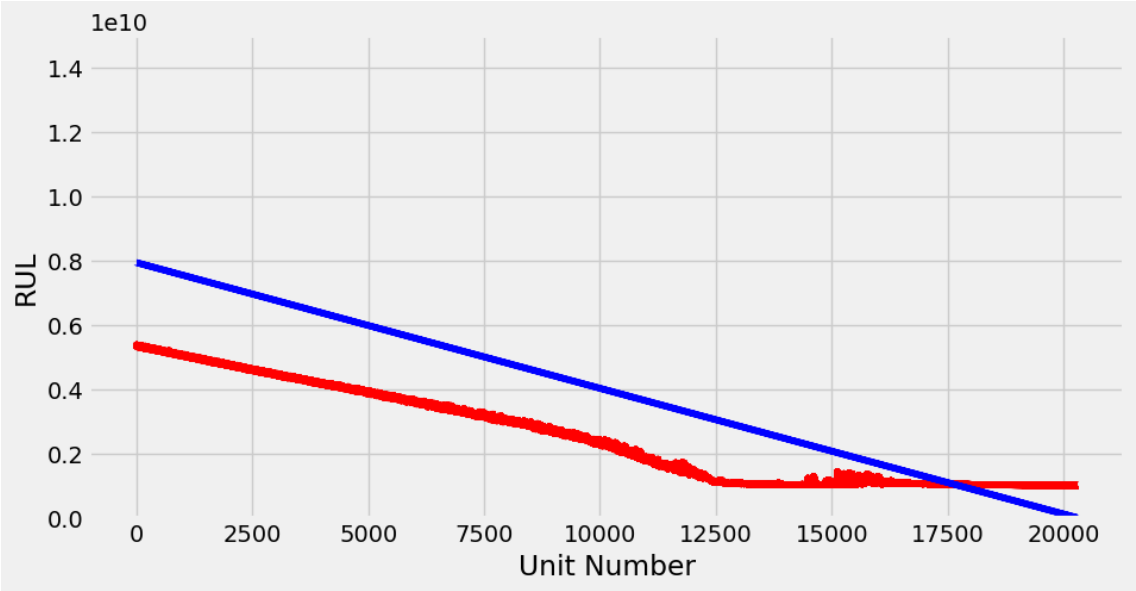

Abbildung 8: RUL-Prediction V.5 Bearing_2_2

KPI`s zur Evaluierung:

- MSE: 0.1874752
- RMSE: 0.4329840
- MAE: 0.3931797
- MAPE: 20190.53%
- Geschätzte RUL: 2:29:18:255000

Besonders fällt auf, dass der MAPE-Wert hoch ist. Wohingegen der MSE-Wert deutlich geringer ist. Insgesamt betrachtet, ist die Vorhersage der RUL von Condition 1 deutlich besser als von Condition 2. Dennoch kann man in dem Ergebnis eine Ähnlichkeit entnehmen.

4.6 Versuch 6: Condition 2 und 3

In diesem Versuch wurden Bearings von Condition 2 und 3 kombiniert, da bei Condition 3 zu wenig Kugellager zu Verfügung waren.

Trainings-Daten: Bearing_3_3, Bearing_2_6, Bearing_2_7

Test-Daten: Bearing_3_1

Des Weiteren wurden folgende Parameter verwendet:

- **Sequenzlänge:** 75
- **Dropout (für alle Layer):** 0,2
- **Epochen:** 4

Ergebnisse:

In der Abbildung ist zu erkennen, dass die vorhergesagte RUL(rot) deutlich ähnlicher zur tatsächlichen RUL (blau) verläuft, als in den Versuchen mit ausschließlich Condition 2.

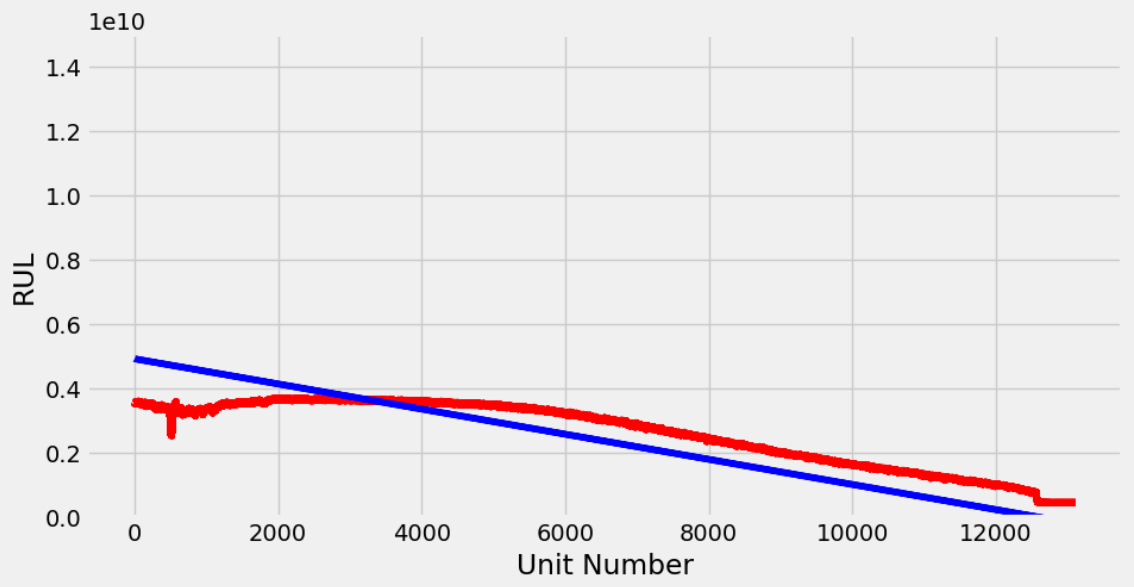

Abbildung 9: RUL-Prediction V.6 Bearing_3_1

KPI`s zur Evaluierung:

- MSE: 0.0720390
- RMSE: 0.2684008
- MAE: 0.2424853
- MAPE: 11506.69%
- Geschätzte RUL: 1:58:59:637000

Die MSE hat ein sehr guter Wert im Vergleich zu allen versuchen. Die RMSE ist wie zu erwarten höher, da die ersten Reihen eine deutlich stärkere Abweichung vorweisen. MAE ist auch in Ordnung wobei die MAPE drastisch hoch liegt, bei der die Aussagekraft in Frage gestellt ist.

5 Fazit

In dieser Arbeit wurde mit einem LSTM-Modell die Remain Usefull Lifetime von Kugellagern (Bearings) prognostiziert.

In den Abbildungen sind die besten Ergebnisse zu den Versuchen der verschiedenen Conditions dargestellt. Dabei ist die vorhergesagte RUL in rot zu sehen und die tatsächliche RUL in blau gezeichnet.

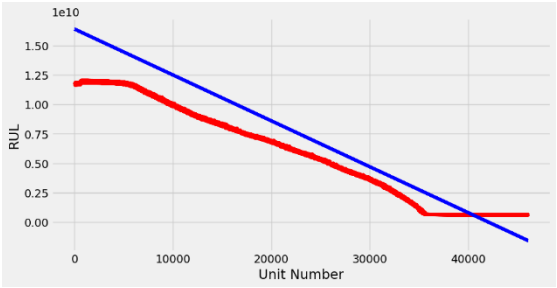

Abbildung 11: Condition 1, Bearing_1_3

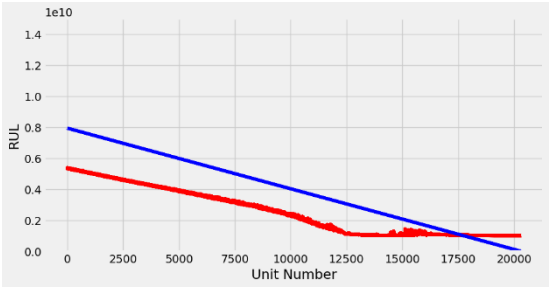

Abbildung 12: Condition 2, Bearing_2_2

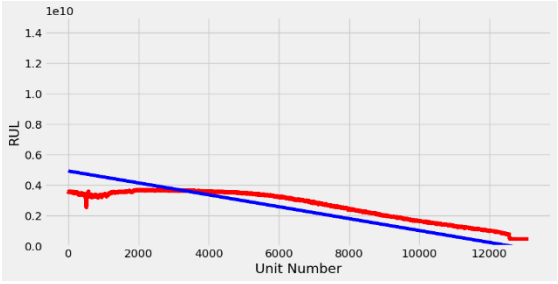

Abbildung 10: Condition 2+3, Bearing_3_1

Dabei ist zu erkennen, dass die prognostizierte RUL meist eine ähnliche Steigung aufweist und die größten Unterschiede bei der größten RUL hat (ganz links in den Schaubildern). Des Weiteren wurden KPI`s zur Evaluierung der Ergebnisse berechnet. Diese sind in der folgenden Tabelle aufgezeigt.

KPI`s	Condition 1	Condition 2	Condition 2+3
MSE	0.0520238	0.1874752	0.0720390
RMSE	0.2280974	0.4329840	0.2684008
MAE	0.2054249	0.3931797	0.2424853
predicted RUL	4:16:13:640000	2:29:18:255000	1:58:59:637000
true RUL	5:33:50:900000	3:12:40:810000	2:22:10:610000

Abbildung 13: KPI`s zu den Testergebnissen

. An den KPI's ist auch zu erkennen dass der Versuch mit Condition1 und der Versuch mit Condition 2+3 deutlich besser abgeschnitten haben als der Versuch

mit der Condition 2. Das liegt vermutlich daran, dass die tatsächliche RUL starke Abweichungen bei der Condition 2 hat.

Die Ergebnisse geben eine gute Tendenz ab, wie die RUL prognostiziert werden kann. Um die Prediction der RUL weiterhin zu optimieren kann anstatt eines LSTM, der Ansatz zur Nutzung eines TSM verwendet werden.

6 References

Adtance.com (2020). Retrieved from https://www.adtance.com/de/blog/2020/die-grossen-verluste-durch-ausfallzeiten-in-produktionsbetrieben#:~:text=Bereits%20der%20einst%C3%BCndige%20ungeplante%20Stillstand,1%2C7%20Millionen%20EUR).

Archd3sai (2020). Predictive-Maintenance-of-Aircraft-Engine/RUL Prediction Regression /LSTM. Retrieved from https://github.com/archd3sai/Predictive-Maintenance-of-Aircraft-Engine/blob/master/RUL%20Prediction%20Regression/LSTM%20RUL%20Prediction.ipynb

Compamind.de (2023). Retrieved from https://compamind.de/knowhow/predictive-maintenance/#:~:text=Predictive%20Maintenance%20eingesetzt%3F-,Was%20ist%20Predictive%20Maintenance%3F,gehalten%20und%20Stillst%C3%A4nde%20vermieden%20werden.

Databasecamp.de (2022). Long Short-Term Memory Networks (LSTM) – einfach erklärt! Retrieved from https://databasecamp.de/ki/lstm

IEEE Reliability Society, FEMTO-ST Institute (2012). IEEE PHM 2012 Prognostic challenge. Retrieved from https://github.com/wkzs111/phm-ieee-2012-data-challenge-dataset/blob/master/IEEEPHM2012-Challenge-Details.pdf

Nicolas Vandeput (2020). Prognose-KPI: RMSE, MAE, MAPE & Bias. Retrieved from https://ichi.pro/de/prognose-kpi-rmse-mae-mape-bias-226247875306541

Valerie Benning (2020). Den Standardfehler des Mittelwertes verstehen und berechnen. Retrieved from https://www.scribbr.de/statistik/standardfehler/