Deutsches Zentrum für Luft- und Raumfahrt
DLR

Ostfalia
Hochschule für angewandte Wissenschaften

Ostfalia
Hochschule für angewandte Wissenschaften

Fakultät Maschinenbau

Institut für Mechatronik

Masterarbeit

Vergleichende Betrachtung von Deep Learning Ansätzen zur automatischen Detektion von Flugmanövern

Verfasser:	Schmalbruch, Patrick
Matr.-Nr.:	70453284
Eingereicht am:	12.07.2024

In Zusammenarbeit mit:
Deutsches Zentrum für Luft- und Raumfahrt e. V., Braunschweig

Erstprüfer:	Prof. Dr.-Ing. Martin Strube
Zweitprüfer:	Prof. Dr.-Ing. Peter Engel
Betriebl. Betreuer:	Robert Schültzky, M. Sc.; Dipl.-Ing. Christina Pätzold

Zusammenfassung

In der vorliegenden Arbeit werden unterschiedliche Ansätze des Deep Learning hinsichtlich der Problemstellung einer automatisierten Erkennung von Flugmanövern miteinander verglichen. Dazu wurden aus dem Stand der Forschung Modellstrukturen für einen ResNet-, einen LSTM- und einen MLP-Ansatz abgeleitet, die anhand von Qualitätskriterien zur Bewertung von Machine-Learning-Modellen einander gegenübergestellt wurden. Zum Training der Modelle wurden Flugdaten aus der Messanlage eines Forschungsflugzeugs des Deutschen Zentrums für Luft- und Raumfahrt verwendet, die im Vorfeld gelabelt, analysiert und aufbereitet wurden. Insgesamt wurden im Zuge dessen 10 Flugmanöver gelabelt. Zu betonen sind der ResNet- und der LSTM-Ansatz, für die jeweils Sensitivitäten von 81,5 % und 91,04 % erzielt werden konnten. Die Tests der Modelle zeigten jedoch, dass die Modelle noch stark überangepasst sind und sich daher noch keiner der betrachteten Ansätze für den Einsatz im Echtzeitbetrieb eignet. Zwar ging aus den Ergebnissen des Vergleichs hervor, dass sich von den drei ausgewählten Ansätzen der LSTM-Ansatz am besten für den Einsatz zur Flugmanöverdetektion eignet. Allerdings zeigten die Ergebnisse ebenso das Potenzial des bisher kaum verwendeten ResNet-Ansatzes für das noch wenig erforschte Feld der Flugmanöverdetektion. Abschließend wurde der Einfluss möglicher Optimierungsansätze für den LSTM-Ansatz erprobt, woraus die Anpassung der Sequenzlänge als vielversprechendste Verbesserungsmaßnahme hervorging.

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte Hilfe angefertigt, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen (explizit auch aus (KI-)Chatverläufen) als solche kenntlich gemacht habe.

Diese Arbeit wurde noch nicht, auch nicht auszugsweise, für eine andere Studien- oder Prüfungsleistung verwendet. Ich bin damit einverstanden, dass die Arbeit einer elektronischen Plagiatsprüfung unterzogen werden kann.

Braunschweig, 12.07.2024

Schmalbruch, Patrick

Danksagung

Hiermit möchte ich mich bei allen Personen bedanken, die mich bei der Anfertigung der Arbeit unterstützt und somit einen maßgeblichen Anteil am Erfolg dieser Arbeit haben. Mein Dank gilt daher meinem Erstprüfer Prof. Dr.-Ing. Martin Strube, der zu jedem Zeitpunkt der Arbeit für Fragen zur Verfügung stand und mir entscheidende Impulse beim Training der Modelle gegeben hat. Außerdem möchte ich Prof Dr.-Ing. Peter Engel danken, dass er sich bereit erklärt hat die vorliegende Arbeit als Zweitprüfer abzunehmen.

Mein besonderer Dank gilt Frau Dipl.-Ing. Christina Pätzold und Herrn M. Sc. Robert Schültzky, die mich über alle Phasen der Arbeit betreut, die Arbeit korrekturgelesen und mir ihr Vertrauen geschenkt haben, dass ich die Arbeit erfolgreich beenden werde. Zudem möchte ich der Einrichtung Flugexperimente und speziell der Arbeitsgruppe Flight Test Instrumentation danken, dass sie mir die Chance gegeben haben, die Arbeit am DLR anfertigen zu können und mich dafür sehr herzlich in ihrem Team aufgenommen haben. Die Zeit dort wird mir stets in positiver Erinnerung bleiben.

Des Weiteren möchte ich den Kollegen des DLR-Instituts für KI-Sicherheit in Ulm - allen voran Herrn M. Sc. Martin Lanz - danken, die zu Fragen im Bereich des Deep Learning immer sehr hilfsbereit waren, die Arbeit ebenfalls korrekturgelesen haben und mir beim Verständnis der Themen sehr weitergeholfen haben.

Mein persönlicher Dank gilt meiner Partnerin Kim Ebeling. Nicht nur, weil sie sich während der Zeit der Masterarbeit für mich zurückgenommen hat. Darüber hinaus konnte ich mich stets auf ihre bedingungslose Unterstützung verlassen, um diese Arbeit erfolgreich abzuschließen. Ich freue mich darauf, diese selbstlose Geste nach Abschluss der Arbeit endlich wieder gebührend wertschätzen zu können.

Ebenso bedanke ich mich bei Drilon Mahaj, dass er sich die Zeit genommen und die Arbeit korrekturgelesen hat und dass er sein Wissen zur Anfertigung der Arbeit freundschaftlich mit mir geteilt hat.

Abschließend möchte ich meiner Familie danken, dass sie mir dabei geholfen haben, es bis zu diesem Punkt in meinem Leben geschafft zu haben und dass sie mich während der Studienzeit stets unterstützt haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADAM *Adaptive Moment Estimation*

ANN *Artificial Neural Networks*

API *Application Programming Interface*

CNN *Convolutional Neural Network*

DigECAT *Digital twin for Engine, Components and Aircraft Technologies*

DigTwin *Digital Twin*

DL *Deep Learning*

DLR *Deutsches Zentrum für Luft- und Raumfahrt*

DNN *Deep Neural Networks*

FCN *Fully Convolutional Network*

FN *False Negatives*

FP *False Positives*

GB *Gigabyte*

GPU *Graphical Power Units*

GUI *Graphical User Interface*

LSTM *Long short-term memory*

MCC *Matthews Correlation Coefficient*

ML *Machine Learning*

MLOps *Machine Learning Operations*

MLP *Multilayer Perzeptron*

netCDF *Network Common Data Form*

ResNet *Residual Network*

RNN *Recurrent Neural Networks*

SGD *Stochastic Gradient Descent*

TN *True Negatives*

TP *True Positives*

twinstash *Digital twins storage and application service hub*

1 Einführung

Zur Bestimmung der Flugeigenschaften eines neuen Flugzeugentwurfs werden im Rahmen der Flugerprobung bestimmte komplexe Manöver geflogen [1]. Anhand der Reaktion eines Flugzeugs kann dabei bestimmt werden, ob dieses für den Luftverkehr zugelassen werden kann oder ob weitere Änderungsmaßnahmen für eine Zulassung notwendig sind [1]. Dieser Prozess muss ebenfalls durchgeführt werden, wenn sich ein Flugzeugentwurf durch die Modifizierung von ursprünglichen Bauteilen oder Baugruppen ändert. Häufig bringen die verschiedenen Messkampagnen und Forschungsarbeiten des Deutschen Zentrums für Luft- und Raumfahrt (DLR) im Bereich der Luftfahrt solche Modifikationen an den Forschungsflugzeugen mit sich [2]. Daher ist eine kontinuierliche Erprobung und Zertifizierung der Flugzeuge notwendig, innerhalb derer verschiedene Flugmanöver im Rahmen des Flugversuchs durchgeführt werden müssen. Damit verbunden sind ein hoher Umwelteinfluss und eine starke Beanspruchung finanzieller und personeller Ressourcen. Aus diesem Grund wird kontinuierlich daran geforscht, die Flugerprobung und Zertifizierung neuer Entwürfe mithilfe digitaler Zwillinge virtuell abwickeln zu können [3]. Digitale Zwillinge sind ein zentrales Konzept in den modernen Ingenieurwissenschaften, das die virtuelle Repräsentation eines physischen Systems oder Prozesses darstellt [4]. Sie erlauben zahlreiche detaillierte Analysen und Simulationen realer Objekte, wodurch sowohl die Effizienz als auch die Präzision in verschiedenen industriellen Anwendungen erheblich gesteigert werden sollen [4]. Der Begriff Digitaler Zwilling beschreibt dabei eine exakte digitale Nachbildung eines physischen Produkts, Systems oder Prozesses [4]. Diese virtuelle Kopie sammelt kontinuierlich Daten über ihren physischen Zwilling mittels Sensoren und vernetzten Technologien. Damit ist es z. B. möglich, den aktuellen Zustand des physischen Objekts in Echtzeit zu überwachen oder bevorstehende Wartungsarbeiten vorherzusagen. Die Relevanz digitaler Zwillinge erstreckt sich bereits über viele verschiedene Fachbereiche. So werden digitale Zwillinge unter anderem in der Fertigungsindustrie, in der Fahrzeugentwicklung oder im Gesundheitswesen eingesetzt, um Prozesse zu optimieren, Kosten zu senken und Systeme zuverlässiger zu machen [5] [6].

1.1 Motivation

Durch die Zusammenführung der Daten aus verschiedenen Quellen wie Sensoren, Wartungsberichten und Flugversuchsprotokollen können digitale Zwillinge genaue Modelle der Flugzustände eines Luftfahrzeugs erstellen. Die daraus gewonnenen Erkenntnisse können genutzt werden, um den Prozess der Systemidentifikation zu beschleunigen, sodass Effizienz und Sicherheit im Flug- und Forschungsbetrieb gesteigert werden. Dabei spielt die Extraktion

von Metadaten aus den vorhandenen Rohdaten eine wichtige Rolle, um die Anforderungen digitaler Zwillinge zu erfüllen. Metadaten können allerdings nicht gemessen werden, sondern geben stattdessen Auskunft darüber, welche Eigenschaften Datensätze oder Objekte besitzen [7]. Dafür müssen sie gegebenenfalls aufwändig errechnet oder durch Fachexperten aus den betroffenen Rohdaten interpretiert werden. Somit ist die Gewinnung von Metadaten eng an das Know-how der Experten eines Fachbereichs geknüpft und zudem mit einem erhöhten Zeitaufwand verbunden. Um dem mithilfe digitaler Zwillinge entgegenzuwirken, gilt es Methoden, Prozeduren und Werkzeuge zur automatisierten Extraktion von Metainformationen aus den Rohdaten zu entwickeln. Diese sollen im Rahmen des Projekts Digital twin for Engine, Components and Aircraft Technologies (DigECAT, dt.: Digitaler Zwilling für Motoren, Komponenten und Luftfahrzeugtechnologien) vom DLR erforscht, umgesetzt und standardisiert werden [8]. Die Projekterzeugnisse sollen in dem sogenannten Digital twins storage and application service hub (*twinstash*) eingesetzt werden [9]. Diese Online-Plattform wird im Rahmen des DigECAT-Projekts entwickelt, um Forschungsflugdaten und Anwendungen zur Weiterverarbeitung von Flugdaten bereitzustellen [9]. Als Teil dieses Vorhabens soll eine automatisierte Identifikation von Flugmanövern auf Grundlage von Rohdaten umgesetzt werden. Mit der größten zivilen Flotte von Forschungsflugzeugen und -hubschraubern betreibt das DLR unter anderem neun verschiedene Flächenflugzeuge, mit denen jeweils teils unterschiedliche Testmanöver durchgeführt werden. Aus diesem Grund sollte der Lösungsansatz für die Problemstellung einer automatisierten Flugmanöverdetektion eine gute Generalisierungsfähigkeit aufweisen. Daher sind speziell Methoden im Bereich Deep Learning (DL, dt.: tiefgehendes Lernen) für diese Aufgabe sehr gut geeignet [10]. Durch die Anwendung von Deep Neural Networks (DNN, dt.: Tiefe Neuronale Netze) ist es im DL-Bereich möglich, hochkomplexe Funktionen abzubilden, die von zahlreichen Einflussfaktoren abhängig sind [11]. Für den Fall einer Manöverdetektion besteht daher das Potenzial, mithilfe von DNNs verschiedene Manöver unterschiedlicher Flugzeuge mit erforderlicher Genauigkeit bestimmen zu können. Mit dem kontinuierlichen Anstieg der Rechenleistung und der verfügbaren Datenmengen des letzten Jahrzehnts hat sich jedoch auch die Vielfalt der potenziell anwendbaren DL-Ansätze stark erhöht [12] [13]. Im Bereich der Signal- und speziell der Flugdatenverarbeitung trifft dies sowohl für die Klassifizierung als auch für die Vorhersage von Signalverläufen zu. Hinzu kommt, dass der Bereich der Klassifizierung von Flugmanövern noch nicht weit erforscht ist [14]. Um ein DL-Modell zur Klassifizierung von Flugmanövern zu entwickeln, muss zunächst also die Frage beantwortet werden, welche existierenden DL-Ansätze im Bereich der Signal- und Flugdatenverarbeitung sich für ein solches Vorhaben eignen. Dafür bedarf es einer umfassenden Analyse potenzieller DL-Ansätze, in der diese

nicht nur hinsichtlich ihrer Genauigkeit, sondern zudem hinsichtlich ihrer Anwendbarkeit im kontinuierlichen Einsatz bewertet werden.

1.2 Aufgabenstellung

Diese Arbeit soll sich mit der folgenden Forschungsfrage beschäftigen:

Welche Deep-Learning-Ansätze eignen sich zur Klassifizierung von Flugmanövern auf Grundlage von Flugmessdaten anhand der Bewertung mit Qualitätskriterien für Machine-Learning-Modelle?

Dafür gilt es zunächst einen Überblick darüber zu verschaffen, welche DL-Ansätze in der Forschung dazu verwendet werden, um entweder multivariate Zeitseriendaten im Allgemeinen oder um Flugmanöver in Flugdaten zu klassifizieren. Zusätzlich sollen aktuelle Qualitätskriterien ermittelt werden, die sich zur Bewertung von Machine-Learning-Modellen eignen. Darauf aufbauend soll eine Literaturrecherche der benötigten Grundlagen zur Umsetzung einer Manöverdetektion durchgeführt werden. Basierend auf dem recherchierten Stand der Forschung gilt es mindestens drei verschiedene DL-Ansätze zur automatisierten Detektion von Flugmanövern abzuleiten, zu trainieren und die Trainingsergebnisse miteinander zu vergleichen. Um diese zu trainieren, müssen potenziell geeignete Trainingsdaten ausgewählt und für das Training vorbereitet bzw. transformiert werden. Zum Vergleich der Ansätze sollen geeignete Metriken identifiziert und angewandt werden, welche die Qualität der Ansätze einerseits hinsichtlich ihrer Leistungsfähigkeit und andererseits hinsichtlich ihrer Gebrauchstauglichkeit im Einsatz beurteilen. Basierend auf den Resultaten der Qualitätsanalyse soll der geeignetste Ansatz identifiziert und im Rahmen einer Experimentierphase weiter erprobt und abschließend erneut evaluiert werden.

2 Stand der Forschung

Zur automatisierten Klassifizierung von Flugmanövern mithilfe von DL werden in der Wissenschaft verschiedene Ansätze zur Problemlösung verfolgt. Einer davon ist die Behandlung als Problem der Klassifizierung multivariater Zeitserien. Allgemein werden dazu häufig DL-Methoden verwendet [15]. Aufgrund der guten Leistungsfähigkeit ist für diese Aufgabe mittlerweile die Anwendung verschiedener Architekturen von eindimensionalen Convolutional Neural Networks (CNN, dt.: Faltendes Neuronales Netz) verbreitet [16]. Ismail Fawaz et al. [16] kamen bei einer vergleichenden Betrachtung zum Schluss, dass dabei vor allem die Netzarchitekturen Fully Convolutional Network (FCN, dt. Vollständig Faltende Netze) und das Residual Neural Network (ResNet, dt.: Residuales Neuronales Netzwerk) herausstechen. Zudem wurde dabei festgestellt, dass sich für die Klassifizierung vereinzelter Datensätze Multilayer Perzeptron (MLP, dt.: Mehrschichtiges Perzeptron) Architekturen ebenfalls gut eignen. Mammeri et al. [17] haben ergänzend dazu die Möglichkeit untersucht FCNs zur automatischen Erkennung von Fahrzeugmanövern mit wenig gelabelten Daten zu trainieren. Dafür wurde eine adaptive halb-überwachte Lernstrategie (engl.: semi-supervised learning) angewandt. Für den speziellen Fall der Vorhersage von Flugmanövern werden FCNs öfters in Kombination mit einer Form der Recurrent Neural Networks (RNN, dt.: Rekurrente Neuronale Netze), den Long Short-Term Memory Netzen (LSTM, dt.: langes Kurzzeitgedächtnis), verwendet [18] [14]. Diese hybriden Strukturen aber ebenso CNNs im Allgemeinen haben sich in mehreren Anwendungsfällen als performante Lösungen bewiesen, die nicht nur eine hohe Leistungsfähigkeit erzielen [17], sondern zudem eine gute Generalisierungsfähigkeit aufweisen [14]. Damit wird eine der Hauptschwächen traditioneller Machine-Learning-Ansätze adressiert [11]. Vereinzelt wird die automatische Detektion von Manövern ebenfalls als ein Bildklassifizierungsproblem behandelt, das mithilfe von 2D-CNNs gelöst wird [19] [20]. Im Bereich der Flugdatenverarbeitung wurden reine LSTM-Netze unter anderem bereits erfolgreich für die Vorhersage von Flugzeugtrajektorien verwendet [21]. Des Weiteren stellten Lu et al. [14] in einer Metastudie Parallelen zwischen dem Problem der Flight Maneuver Recognition (dt.: Erkennung von Flugmanövern) und dem der Human Action Recognition (dt.: Erkennung menschlicher Handlungen). In diesem Bereich der Mustererkennung verwenden Tchuente et al. unter anderem ein MLP zur Erkennung von aggressiven menschlichen Handlungen [22]. Der verwendete Datensatz wird dabei vorab in fortlaufende Datensequenzen unterteilt, aus denen physikalische Merkmale extrahiert und an das MLP als Eingangsdaten übergeben werden.

Zum Vergleich verschiedener DL-Architekturen für die Klassifizierung von Daten ist eine Methode zur Bewertung der Modellqualität notwendig. Siebert et al. stellen dazu ein

multidisziplinäres Qualitätsmodell bereit, welches basierend auf Normen wie z. B. den ISO/IEC-Normen 25010 und 8000 entwickelt wurde und mitunter die Bewertung von Machine Learning-Modellen im Allgemeinen standardisieren soll [23]. In dem Bereich der digitalen Medizin wurde das Modell z. B. bereits zur Bewertung von DL-Ansätzen angewandt [24].

3 Theoretische Grundlagen

Flugmanöver werden durch Steuereingaben des Piloten initiiert, die einen vorherrschenden Flugzustand stören. Die daraus resultierende Reaktion des Flugzeugs ist im Nachhinein in den entsprechenden Sequenzen der aufgenommenen Signalverläufe erkennbar. Um die Sequenzen den jeweiligen Flugmanövern zuzuschreiben, ist eine Interpretation der Signalverläufe notwendig. Dazu wird mitunter das Wissen darüber benötigt, wie sich verschiedene Manöver in bestimmten Signalverläufen niederschlagen. Diese Aufgabe kann, unter der Verwendung von Deep Learning Algorithmen, mithilfe einer automatisierten Manövererkennung realisiert werden. Dazu werden im Folgenden die benötigten Grundlagen über die Flugmechanik von Flächenflugzeugen, die Flugerprobung und die betroffenen Flugmanöver sowie über DNNs behandelt. Abschließend wird der *twinstash* kurz vorgestellt, auf dem im Rahmen des DigECAT-Projekts ein geeigneter DL-Ansatz zur Detektion von Flugmanövern in die Anwendung gebracht werden soll.

3.1 Flugmechanik von Flächenflugzeugen

Bei einer detaillierten Betrachtung von Flugmanövern ist es sinnvoll die Hintergründe der Flugmechanik eines Flächenflugzeugs (im Folgenden kurz: Flugzeug) zu verstehen. So wird ein besseres Verständnis über die charakteristischen Eigenschaften und den Zweck eines Manövers geschaffen. Speziell zu Beginn einer Erprobungsphase werden Manöver geflogen, um als Reaktion eine bestimmte Antwort des Flugzeugs zu provozieren. Daher befasst sich das folgende Unterkapitel mit den Teilgebieten der Flugmechanik, in denen die Steuerbarkeit und Stabilität eines Flugzeugs beschrieben werden. Dabei wird sich auf diejenigen Aspekte beschränkt, welche das notwendige Hintergrundwissen für die Manöver vermittelt, die in dieser Arbeit behandelt werden. Dazu zählen Teile der statischen Stabilität sowie der Steuerung von Längs- und Seitenbewegungen eines Flugzeugs. Zur grundlegenden Orientierung werden vorab kurz der Aufbau und die Hauptkomponenten von Flugzeugen erläutert.

3.1.1 Aufbau und Hauptkomponenten

Aufgrund der Relevanz für diese Arbeit, nachfolgend kurz wichtige Spezifikationen des Forschungsflugzeugs Cessna Grand Caravan 208B (im Folgenden kurz: Cessna) erläutert. Es ist Teil der Forschungsflotte des DLR und dient im Rahmen dieser Arbeit als Quelle für die Daten zum Training der DL-Algorithmen. Im Fall der Cessna handelt es sich um ein einmotorig angetriebenes Eindecker-Flugzeug mit einem verstellbaren dreiblättrigen Propellerantrieb.

Tabelle 3-1: Technische Spezifikationen DLR Cessna Grand Caravan 208B (D-FDLR) [25]

Spezifikationen	Daten
Länge	12,7 m
Höhe	4,57 m
Spannweite	15,9 m
Kabinenlänge	4,82 m
Kabinenbreite	1,63 m
Kabinenhöhe	1,37 m
Sitzplätze	13
Leergewicht	2,3 t
Max. Abflugmasse	3,96 t
Antrieb	Pratt & Whitney Canada Triebwerk, Modell PT6A-114 mit 675 Wellen-PS
Propeller	dreiblättriger, verstellbarer Hartzell-Propeller
Reichweite	1660 km
Max. Flughöhe	7620 m (25.000 ft)
Max. Geschwindigkeit	314 km/h (10.000 ft)
Flugdauer	5:30 Std. ohne Nutzlast
Tankkapazität	1 t
Nutzungsprofil	Passagier-, Rettungs-, Versorgungs- und Frachtflugzeug
DLR-Flugbetrieb	Oberpfaffenhofen

Tabelle 3-1 fasst ergänzend dazu die wichtigsten technischen Spezifikationen zusammen. Zu betonen sind hierbei die maximal erzielbare Flughöhe von 7.620 m und die maximal erzielbare Geschwindigkeit von 314 km/h. Zusätzlich wurden am DLR zahlreiche Modifikationen an dem Flugzeug vorgenommen, sodass es sich beispielsweise für Fernerkundungsmissionen eignet

oder im Rahmen der „DLR Summer School“ als sogenannter fliegender Hörsaal dient (Abbildung 3-1). [25] [26]

Abbildung 3-1: Modifikationen DLR Cessna Grand Caravan 208B (D-FDLR) [26]

Die Cessna ist zudem mit zusätzlichem Messequipment zur Lagewinkelbestimmung, Positionsbestimmung in Echtzeit, Datenvisualisierung in Echtzeit, Druckmessung, Temperaturmessung und Feuchtigkeitsmessung ausgestattet. Eine Auflistung der Messsignale (im Folgenden auch: Parameter oder Features), welche die Cessna aufnimmt ist der Arbeit in Anhang A1 beigefügt.

Eine der wesentlichen Aufgaben der Flugführung ist es, einen angestrebten Flugzustand zu erreichen und beizubehalten. Zur möglichst präzisen Definition eines Flugzustands werden verschiedene Informationen über die Umgebung sowie das Luftfahrzeug selbst benötigt. Um seinen aktuellen Flugzustand erfassen zu können, ist ein Flugzeug daher mit Sensorik ausgestattet, welche die erforderlichen Größen entweder direkt oder rechnerisch mithilfe von sogenannten Ersatzmessgrößen ermitteln kann. Direkt messbar ist zum Beispiel die Temperatur. Dagegen kann beispielsweise die Geschwindigkeit relativ zur Luft lediglich unter Berücksichtigung des messbaren Luftdrucks berechnet werden. Die Messgrößen werden mit verschiedenen und teils mehrfachen Ausführungen von unter anderem Druckmessgeräten, Windfahnen und Beschleunigungssensoren ermittelt. Außerdem sind Flugzeuge zur Positionsbestimmung mit Navigationssystemen ausgestattet. Hierfür kommen verschiedene Navigationsverfahren zum Einsatz, auf die im Rahmen dieser Arbeit nicht näher eingegangen wird. Im Bereich der Erprobung kommt es zudem häufig zum Einsatz zusätzlicher Messausstattung, mit dessen Daten Zielparameter genauer oder in einer höheren Anzahl erfasst werden können. [27]

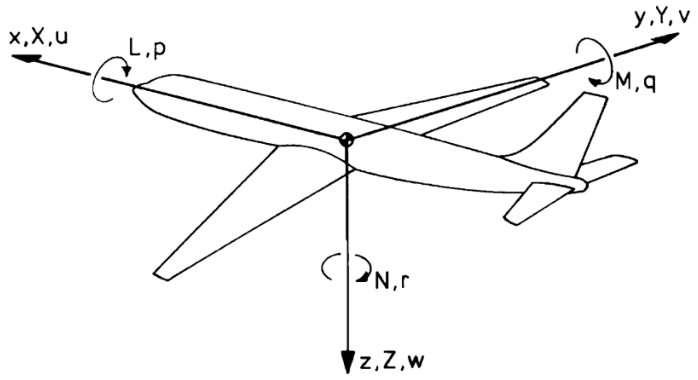


Abbildung 3-2: Kräfte und Momente im flugzeugfesten Koordinatensystem [28]

Die Bewegung eines Flugzeugs kann vom Piloten durch die Änderung von Triebwerksschub und Ruderwinkeln beeinflusst werden. Zur Beschreibung der Komponenten vektorieller Kräfte und Momente, die sich dabei ständig ändern, werden in der Luftfahrt unterschiedliche Koordinatensysteme verwendet. Zur besseren Orientierung wird hier zunächst das in Abbildung 3-2 dargestellte flugzeugfeste Koordinatensystem näher erläutert. Grundlegend dient dafür ein rechtshändisches orthogonales Koordinatensystem mit der Längsachse x, der Querachse y und der Hochachse z, wobei die z-Achse nach Flugzeugunten zeigt. Die Kraftkomponenten werden durch X, Y und Z und die Geschwindigkeitskomponenten durch u, v und w in den jeweiligen Richtungen repräsentiert. Bei den Geschwindigkeitsangaben eines Flächenflugzeugs wird zwischen der Absolutgeschwindigkeit verglichen zur Erde V_K, der Relativgeschwindigkeit verglichen zur Luft V_A und der Relativgeschwindigkeit des Windes verglichen zur Erde V_W unterschieden. V_K ist dabei stets die Summe aus V_W und V_A. Hinzu kommen sogenannte Roll-, Nick- und Giermomente L, M, N um die x-, y- und z-Achse, die durch Ausschläge von Quer-, Höhen- und Seitenruder ξ, η, ζ hervorgerufen werden können. Diese verändern wiederum die Roll-, Nick- und Gierwinkel Φ, Θ, Ψ und damit die Lage des Flugzeugs im Raum. Die Betätigung der Quer- und Höhenruder erfolgt über das Drehen sowie das Drücken und Ziehen des Steuerhorns oder Steuerknüppels durch den Piloten. Der Seitenruderwinkel kann über die Betätigung der Pedalerie durch den Piloten beeinflusst werden. Die Flugzeuglagewinkel werden zwischen dem flugzeugfesten Koordinatensystem und einem weiteren, dem erdfesten oder auch geodätischen Koordinatensystem, aufgetragen. Die z-Achse des erdfesten Koordinatensystems wird anhand der Wirkrichtung der Gewichtskraft definiert. Die horizontale x-y-Ebene liegt dementsprechend parallel zur Erdoberfläche, die näherungsweise als flach angesehen wird. Die x-Achse wird dabei üblicherweise durch den magnetischen Norden definiert. Ergänzend dazu sind die Flugzeuglagewinkel in Abbildung 3-3 dargestellt. Die Achsen des erdfesten

Koordinatensystems sind mit dem Index g und die des flugzeugfesten Koordinatensystems mit dem Index f markiert [28]

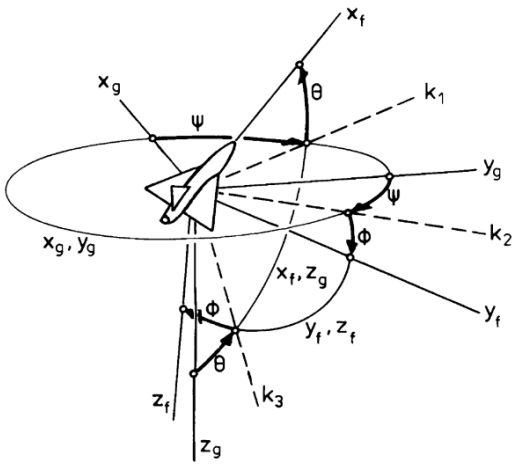


Abbildung 3-3: Lagewinkel anhand des flugzeugfesten Koordinatensystems verglichen zum erdfesten Koordinatensystem [28]

Zudem sind zwei weitere wichtige Achsensysteme zu nennen: das aerodynamische und das experimentelle Achsensystem. Das aerodynamische Achsensystem zeichnet sich dadurch aus, dass sich dessen Längsachse x_a an der Richtung des Geschwindigkeitsvektors des Flugzeugs orientiert. Die Hochachse z_a liegt zudem auf der z-Achse des experimentellen Achsensystems z_e. Diese wird wiederum durch die Längsachse x_e und die Querachse y_e des experimentellen Achsensystems definiert. x_e ist dabei gleich der Projektion des Geschwindigkeitsvektors auf die x-y-Ebene des flugzeugfesten Koordinatensystems und y_e entspricht der Querachse des flugzeugfesten Koordinatensystems y_f. Die Wichtigkeit der beiden Achsensysteme kommt daher, dass anhand dessen der Anstellwinkel α (Winkel zwischen x_e und x_f) und der Schiebewinkel β (Winkel zwischen x_e und x_a) abgeleitet werden können (Abbildung 3-4). [27]

Anhand der oben beschriebenen Größen können Flugzeugbewegung und -zustand definiert werden. Die Bewegungen des Flugzeugs werden dabei in Längs- und Seitenbewegungen unterteilt. Das Flugzeugverhalten bei diesen Bewegungsformen kann anhand der Analyse der Flugzeugreaktion auf bestimmte Flugmanöver ermittelt werden.

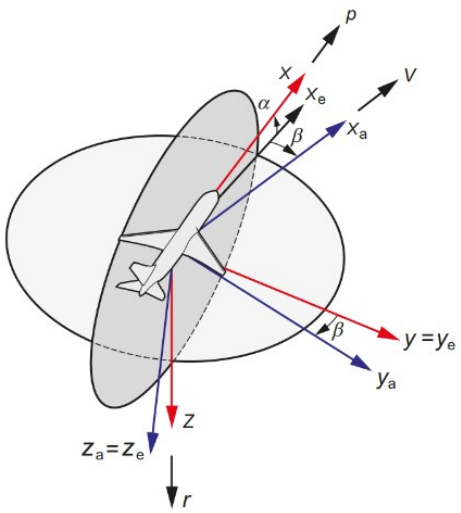


Abbildung 3-4: Experimentelles und aerodynamisches Achsensystem [27]

Im folgenden Abschnitt werden dazu die flugmechanischen Aspekte der Längs- und Seitenbewegung kurz beleuchtet, um ein besseres Verständnis für die Durchführung bestimmter Manöver zu vermitteln.

3.1.2 Längsbewegung

Die Längsbewegung findet isoliert in der vertikalen x-z-Ebene statt. Die Betrachtungen gehen von einem (quasi-)stationären Flugzustand aus, in dem sich die Zustandsgrößen des Flugzeugs kaum oder gar nicht verändern. Daher wird dabei auch vom stationären Geradeausflug gesprochen. Dies umfasst neben Steig-, Sink- und Horizontalflügen auch kreisförmige Abfangbewegungen in der x-z-Ebene. Letztere können dabei durch vereinfachte Annahmen mitberücksichtigt werden und liefern wichtige Informationen über die Flugeigenschaften. [29]

Steuerung

Die Steuerung der Längsbewegung erfolgt über die Änderung des Triebwerkschubs F und des Höhenruderwinkels η durch den Piloten. Damit verbunden sind Steuerkräfte, denen eine hohe Bedeutung für die Flugeigenschaften zugesprochen werden. Zum einen sind die Steuerkraft zur Gewährleistung des stationären Geradeausflugs sowie die Stabilitätshandkraft für die Bestimmung der Flugeigenschaften von erhöhtem Interesse. Im Zusammenhang mit der Stabilitätshandkraft spielt der sogenannte Trimmzustand (auch: ausgetrimmter Flugzustand) eine wichtige Rolle. Dieser beschreibt den Zustand eines Flugzeugs, der die Stabilitätshandkraft eliminiert und dient hauptsächlich der Entlastung des Piloten. Zum anderen ist die Steuerkraft zur Stabilisierung einer Abfangbewegung in der x-z-Ebene, die Abfanghandkraft, eine wichtige Messgröße in der Flugzeugentwicklung. Dafür verantwortlich

sind gesetzliche Vorgaben für minimal und maximal zulässige Abfanghandkräfte. Für diese ist es z. B. wichtig zu erwähnen, dass sie gemeinsam mit der Flächenbelastung des Flugzeugs anwachsen. [27]

Stabilität

Ein Flugzeug wird als stabil bezeichnet, wenn ein aktueller Flugzustand, der durch eine Störung unterbrochen wird, sich daraufhin ohne Eingreifen des Piloten wiedereinstellt. Eine solche Störung kann beispielsweise durch äußere Einflüsse, wie z. B. Wetterverhältnisse, oder Eingriffe des Piloten in die Flugsteuerung hervorgerufen werden. Die Flugzeugstabilität nimmt eine wichtige Rolle im Rahmen der Flugeigenschaftsvermessung, die in Kap. 3.2 detaillierter erläutert wird. Der Begriff der Stabilität unterteilt sich in der Luftfahrt weiter in die statische und dynamische Stabilität. Im Rahmen dieser Arbeit wird lediglich die statische Stabilität der Längs- und Seitenbewegung näher betrachtet. Diese liegt vor, wenn aus dem Auftreten einer Störung eine Rückstellkraft bzw. ein Rückstellmoment resultiert. [29]

Die Eigenschaft der statischen Stabilität bei der Längsbewegung ist stark abhängig von der Schwerpunktlage und impliziert, dass die statischen Momente einer Abweichung vom Gleichgewichtszustand entgegenwirken [27]. Dementsprechend wird durch die statische Stabilität eines Flugzeugs der zulässige Schwerpunktbereich maßgeblich beeinflusst [27]. Speziell in Verbindung mit der Längsbewegung nimmt die statische Stabilität deswegen einen hohen Stellenwert ein. Hier sind vor allem der wirkende Gesamtauftrieb und Gesamtwiderstand von Interesse. Der Gesamtwiderstand setzt sich aus Einzelwiderständen zusammen, „die an den Baukomponenten des Flugzeugs entstehen […], getrennt nach Null- und Auftriebswiderstand“ [27]. Die Auftragung des Gesamtauftriebs C_A über dem Gesamtwiderstand C_W in Beiwertschreibweise wird als Flugzeugpolare bezeichnet und ist in Abbildung 3-5 dargestellt. Dabei ist der Gesamtwiderstand auf der Horizontalachse und der Gesamtauftrieb auf der Vertikalachse aufgetragen, während der Anstellwinkel auf dem Verlauf der Flugzeugpolare selbst aufgetragen ist. Der Verlauf kann näherungsweise mit der folgenden quadratischen Funktion beschrieben werden:

$$C_W = C_{Wmin} + k * (C_A - C_{A0})^2 \qquad 3.1$$

Wie in Abbildung 3-5 zu erkennen ist, weicht der wahre Verlauf der Polare mit steigendem Auftrieb von der quadratischen Funktion in Gl. 3.1 ab. Ergänzend wird häufig auch die Auftragung des Auftriebs in Beiwertschreibweise über dem Anstellwinkel α angegeben. In diesem Fall sind die Verläufe mit verschiedenen Klappenwinkeln angegeben, worauf im weiteren Verlauf der Arbeit jedoch nicht näher eingegangen wird.

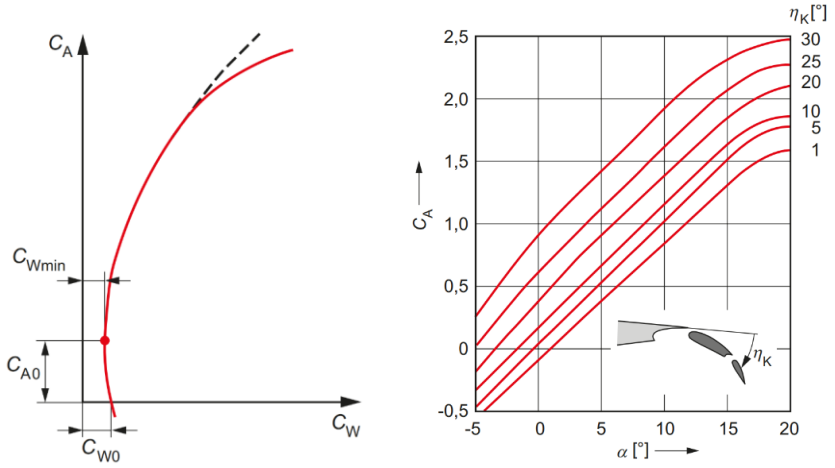


Abbildung 3-5: Flugzeugpolare und Auftriebskennlinie bei verschiedenen Klappeneinstellungen [27]

Für den Auftrieb eines Flugzeugs tragen sowohl Flügel- und Rumpfstruktur als auch das Höhenleitwerk bei. Für eine detaillierte Betrachtung der dabei wirkenden Kräfte ist es deswegen üblich, ein Flugzeug in die zwei Teilsysteme Flügel-Rumpf-Kombination und Höhenleitwerk aufzuteilen (Abbildung 3-6). [27]

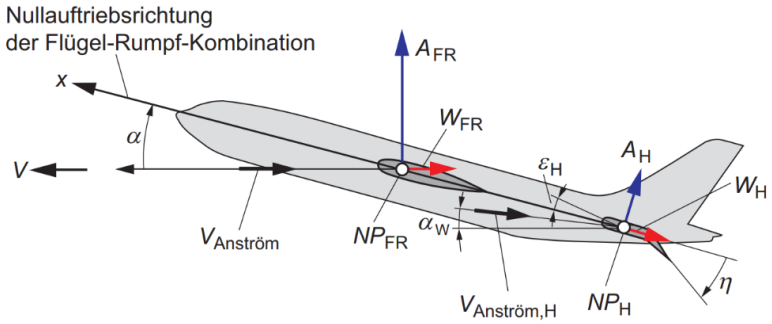


Abbildung 3-6: Kräfte, Koordinatensysteme und Anströmwinkel [27]

Ausschlaggebend für die Flugzeugstabilität sind vor allem die Antworten des Flugzeugs auf Störungen bzw. die damit verbundenen Auftriebsänderungen. Beispielsweise bedeutet eine Anstellwinkeländerung zugleich eine Änderung der Teilauftriebe von Flügel-Rumpf-Kombination und Höhenleitwerk, was wiederum Nickmomente hervorruft, die auf das gesamte Flugzeug einwirken. In diesem Zusammenhang ist die hohe Bedeutung des sogenannten Neutralpunktes zu nennen. Der Neutralpunkt beschreibt hierfür den ortsfesten Punkt eines Flugzeugs (definiert durch die Konfiguration), in dem sich diese Nickmomente infolge einer Anstellwinkeländerung gegenseitig aufheben. Dementsprechend dient der Neutralpunkt als Angriffspunkt für den im Zuge dessen hervorgerufenen Gesamtauftrieb aufgrund einer Anstellwinkeländerung. Der starke Zusammenhang zwischen Neutralpunkt und statischer

Stabilität lässt sich an dieser Stelle mit dem örtlichen Verhältnis zum Schwerpunkt erklären. [27]

Dafür werden drei Fälle unterschieden:

Statisch stabil: Schwerpunkt liegt vor dem Neutralpunkt, sodass das Nickmoment infolge einer Auftriebsänderung der Störung entgegenwirkt

Neutral stabil: Schwerpunkt und Neutralpunkt sind deckungsgleich, sodass kein Nickmoment infolge einer Auftriebsänderung entstehen kann

Statisch instabil: Schwerpunkt liegt hinter dem Neutralpunkt, sodass das Nickmoment infolge einer Auftriebsänderung die Störung verstärkt

Einen weiteren wichtigen Aspekt der Stabilität stellen die Abfang- oder Manöverstabilität und der damit verbundene Manöverpunkt dar [27] [30]. Anhand der Manöverstabilität werden alle Kräfte und Momente definiert, die im Zusammenhang mit einer Abfangbewegung in der x-z- oder y-x-Ebene auftreten. Dafür wird von einer kreisförmigen Bewegung in der jeweiligen Ebene ausgegangen. Der Manöverpunkt beschreibt dabei die am weitesten rückwärtige Schwerpunktlage x_S, bei welcher die zum Durchfliegen einer Kreisbahn notwendige Runderwinkeländerung $\Delta\eta = 0$ ergibt.

3.1.3 Seitenbewegung

Anders als bei der Längsbewegung handelt es sich bei der Seitenbewegung eines Flugzeugs (auch: unsymmetrischer Flug [28]) um eine nicht isolierte Bewegung. Die Seitenbewegung eines Flugzeugs kann durch drei Bewegungskomponenten beschrieben werden: den Schiebewinkel β, die Roll-Drehgeschwindigkeit p und die Gier-Drehgeschwindigkeit r [27]. Daran zeigt sich bereits, dass die Seitenbewegung eines Flugzeugs aus der Gier- und der Rollbewegung entsteht. Analog zur Längsbewegung kann der Begriff der Stabilität hier ebenso in statische und dynamische Stabilität unterteilt werden. Die statische Stabilität wird in diesem Fall jedoch üblicherweise als Windfahnen- oder Richtungsstabilität bezeichnet [27].

Steuerung

Zur Steuerung der Seitenbewegung muss grundsätzlich eine Kraft in y-Richtung erzeugt werden. Bei einer Rollbewegung wird dazu in der Regel die y-Komponente des Auftriebsvektors verwendet. Die Rollbewegung wird dazu mit der Betätigung der Querruder gesteuert. Dies wird als wirksamste Methode angesehen, da hierbei die notwendige Ruderkraft im Verhältnis zum Betrag der y-Komponente des Auftriebs sehr klein ist. Dafür wird allerdings eine verzögerte Flugzeugreaktion in Kauf genommen. Als Konsequenz sinkt die z-

Komponente der Auftriebskraft, sodass durch alleiniges Rollen des Flugzeugs das Gleichgewicht zwischen Gewichtskraft und Auftrieb gestört wird. Aus diesem Grund ist es notwendig zusätzlich das Höhenruder auszuschlagen, um einen Höhenverlust während des Kurvenflugs zu vermeiden. So kann die Auftriebskraft und damit auch dessen Komponenten erhöht werden. Dies hat ebenfalls einen Anstieg des Lastfaktors während des Kurvenflugs zur Folge. Im Kurvenflug wird zudem das Seitenruder betätigt, womit die Gierbewegung eines Flugzeugs gesteuert wird. Dies ist notwendig, da das Rollen eines Flugzeugs ebenfalls eine Gierbewegung hervorruft und dadurch ungewollte Schiebewinkel vermieden werden sollen. [27]

Stabilität

Die statische Stabilität der Gierbewegung ist dadurch gegeben, dass bei auftretendem Schiebewinkel β am Seitenruder des Flugzeughecks eine entgegengesetzte Seitenkraft erzeugt wird, die für ein stabilisierendes Giermoment sorgt. Die Flügel-Rumpf-Kombination trägt dabei ebenfalls zur Querkraft bei. Da in fast ausnahmslos allen Flugzeugkonfigurationen das Seitenruder über der Flugzeuglängsachse zu verorten ist, erzeugt die dort infolge von β wirkende Querkraft Momente um x und y. Da durch eine Rollbewegung keine weiteren aerodynamischen Kräfte und Momente direkt erzeugt werden, handelt es sich dabei auch nicht um eine stabile Bewegung. An dieser Stelle ist es wichtig anzumerken, dass zwischen Gier- und Rollbewegung mehrere Wechselwirkungen entstehen. Ein wichtiger Aspekt ist in diesem Zusammenhang das Schiebe-Rollmoment, welches erhöhten Einfluss auf die Beurteilung der Kurvenflugeigenschaften nimmt. Durch einen erhöhten Schiebewinkel β werden die beiden Tragflächen unterschiedlich stark angeströmt. Daraus resultieren unterschiedliche Auftriebskräfte an beiden Tragflächen, wodurch eine örtliche Anstellwinkeländerung an beiden Flügeln stattfindet. Dies führt in der Konsequenz zu einem Rollmoment: dem Schiebe-Rollmoment. Dementsprechend verhält sich dieses Rollmoment proportional zur Auftriebsdifferenz zwischen linkem und rechtem Flügel. Neben diesem treten bei der Seitenbewegung noch weitere Kopplungseffekte auf. Dazu zählen das Gier-Rollmoment, das Seitenruder Rollmoment, das Roll-Giermoment sowie das Querruder Giermoment. Diese werden im Folgenden zwar nicht detaillierter beleuchtet, verdeutlichen jedoch, dass zwischen Gier- und Rollbewegung starke Wechselwirkungen auftreten, was sich auch in den Messsignalen niederschlägt. [27]

3.2 Flugerprobung

Zur Begriffsdefinition und zu den Hintergründen der Flugerprobung (dt.: Flight Test Engineering) der zivilen Luftfahrt liefert z. B. Stoliker umfassende Ausführungen [1]. Ziele des

Flight Test Engineering sind demnach vielfältig. Einerseits können Untersuchungen neuartiger Konzepte oder die Bestätigung bzw. der Widerspruch von Designannahmen im Fokus stehen. Andererseits kann Kern eines Testverfahrens auch die Verifizierung der Leistungsfähigkeit eines Luftfahrzeugs sein. Gemeinsam haben dabei laut Stoliker alle Testverfahren, dass sie vorherige Annahmen, Berechnungen oder Simulationen auf Ihre Allgemeingültigkeit hin überprüfen und diese gegebenenfalls bestätigen oder korrigieren müssen. Die Art der durchzuführenden Testverfahren unterscheidet sich demzufolge maßgeblich aufgrund von zwei Faktoren: der jeweiligen Erprobungsphase und der ausführenden Institution. Die Erprobungsphase gliedert sich nach Stoliker weiter in die Entwicklung und die Zertifizierung auf. Während der Entwicklung werden hauptsächlich Testverfahren durchgeführt, um den Entwicklungsstand eines Fluggeräts hin zu einem Reifegrad voranzutreiben, der einer vorab definierten Anforderungsliste genügt. Dahingegen wird mit der Zertifizierung die Erfüllung dieser Anforderungsliste unter Berücksichtigung der geltenden Gesetzesvorgaben bestätigt. Durch die Tätigkeitsfelder der Forschung und Zertifizierung nimmt das DLR in diesem Zusammenhang die Rolle des Herstellers bis inkl. der Erprobung ein. Dabei werden die Kompetenzen in Bereichen der Weiterentwicklung von Baugruppen und deren Prozessen sowie im Bereich der Erforschung zukunftsfähiger Technologien eingesetzt. Dies erfolgt im „Verbund mit der Industrie“, um „Schlüsseltechnologien für eine umweltschonende und effiziente Luftfahrt“ [31] zu erproben und in die Anwendung bringen zu können [31].

Details des Erprobungsprozesses werden im folgenden Abschnitt unter dem Leitbegriff der Flugeigenschaftsvermessung behandelt.

3.2.1 Flight Envelope

Eine der Hauptdisziplinen des Flight Test Engineering ist es, die Eigenschaften eines Flugzeuges zu messen und zu bestimmen. Die sogenannte Flight Envelope (dt.: Flugbereich, im Folgenden kurz: Envelope) dient dazu als maßgebliches Instrument. Die Envelope wird nach Stoliker in der Erprobung dazu verwendet, um die Flugbereiche eines Flugzeugs zu beschreiben, innerhalb derer eine sichere Flugzeugsteuerung möglich und außerhalb derer ein Flug sicherheitstechnisch inakzeptabel ist [1]. Der Begriff der Flugbereiche bezieht sich hierbei auf verschiedene Parameterbeziehungen, deren Grenzen ermittelt werden müssen. Dabei existieren mehrere Beziehungen, die jeweils unterschiedliche Informationen über die Flugzeugeigenschaften liefern. Als Beispiel dient an dieser Stelle das Höhe-Geschwindigkeit-Verhältnis, das laut Rossow unter anderem für verschiedene Schwerpunkte oder unterschiedliche Klappenstellungen des Flugzeugs ermittelt werden muss [27]. Dieses wird

durch die aerodynamische Grenze, die Leistungsgrenze, die Strukturgrenze und die Temperaturgrenze eingeschränkt (Abbildung 3-7) [27].

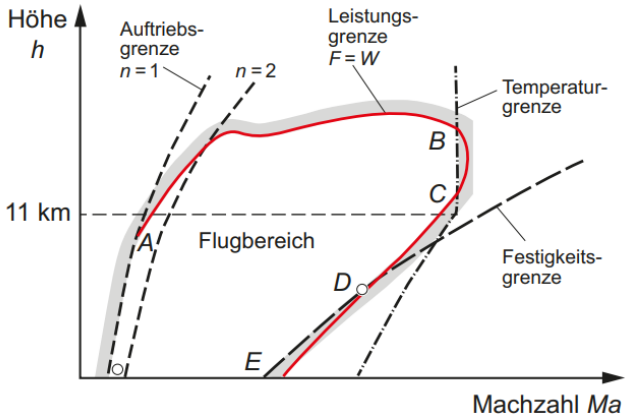


Abbildung 3-7: Flugbereiche beschränkt durch Manövergrenzen [27]

Aus Abbildung 3-7 wird deutlich, dass niedrige Geschwindigkeitsbereiche durch die aerodynamische Grenze (hier: Auftriebsgrenze) definiert werden. Aufgrund des mit zunehmender Höhe sinkenden Leistungsvermögen luftatmender Antriebe, schließt an die aerodynamische Grenze die Leistungsgrenze an. Mit hoher Geschwindigkeit wächst aufgrund des dynamischen Drucks auch die mechanische Belastung auf die Außenstrukturen des Flugzeugs an. Daraus resultiert die Strukturgrenze (in Abbildung 3-7 als Festigkeitsgrenze bezeichnet). Da sich die Temperaturgrenze nur auf Überschallgeschwindigkeiten bezieht, wird diese Grenze nicht weiter thematisiert. [27]

Um die Envelope zu bestimmen, sind mit dem aktuellen Stand der Technik Flugversuche im Rahmen der Flugerprobung unabdingbar. Bevor diese jedoch durchgeführt werden können, sind aus Sicherheitsgründen zahlreiche Design-Reviews, Simulationen und Analysen durch Labortests notwendig. Anhand dessen können dann erste Vorhersagen zum Flugverhalten unter vielfältigen Bedingungen getroffen werden. Doch auch in der Phase des Flugversuches werden die Grenzen der Envelope zunächst nicht ausgereizt, sodass Geschwindigkeiten anfangs noch stark begrenzt und Komplexitäten der Manöver noch stark reduziert werden. Sollten die zugrunde liegenden Vorhersagen anhand der ersten Flugversuche bestätigt werden können, wird die Envelope daraufhin Schritt für Schritt erweitert bis dessen Grenze gefunden wird. Im Testverlauf halten parallel immer komplexere Manöver Einzug in die Testphase, die im folgenden Abschnitt anhand einiger Beispielmanöver detailliert betrachtet werden.

3.2.2 Methoden der Flugerprobung

In Verbindung mit der Flugerprobung wird häufig von der Erfassung der Handling- und Flugeigenschaften eines Flugzeugs gesprochen [27] [1] [32]. Dem Erforschungsprozess der Envelope entsprechend können auch die dafür zu fliegenden Manöver unterteilt werden. Dafür liefern Wichmann et al. eine sogenannte Manöver Pyramide, welche die Testphase hinsichtlich der zu fliegenden Manöver strukturiert (Abbildung 3-8) [33]. Die Testphase wird dabei in eine Open-Loop-Phase (Phase 1), eine Closed-Loop-Phase (Phase 2) und eine Operational-Phase (Phase 3) unterteilt. Je fortgeschrittener hierbei die Designphase ist, desto schwieriger und kostenaufwändiger sind Designänderungen am Flugzeug umzusetzen. Aus diesem Grund werden speziell in den Phasen 1 und 2 hohe Anstrengungen unternommen, um die Handling- und Flugeigenschaften bestmöglich zu untersuchen und zu optimieren. Aufgrund der Anwendung in den allerersten Flügen der gesamten Testphase, dienen die Open-Loop-Manöver der Phase 1 zunächst dazu, die Testpiloten mit den grundlegenden Flugzeugeigenschaften vertraut zu machen. Zu diesem Zweck wird vorerst das Flugzeug dynamisch angeregt und dessen Antwort auf die Anregungen beobachtet. Wie bereits zum Ende von Kapitel 3.2.1 angedeutet, handelt es sich lediglich um risikoarme Manöver, die in dieser Phase durchgeführt werden. Für die vorliegende Arbeit werden vorrangig Manöver der Open-Loop-Phase behandelt, weshalb auf die übrigen Phasen nicht weiter eingegangen wird. [32]

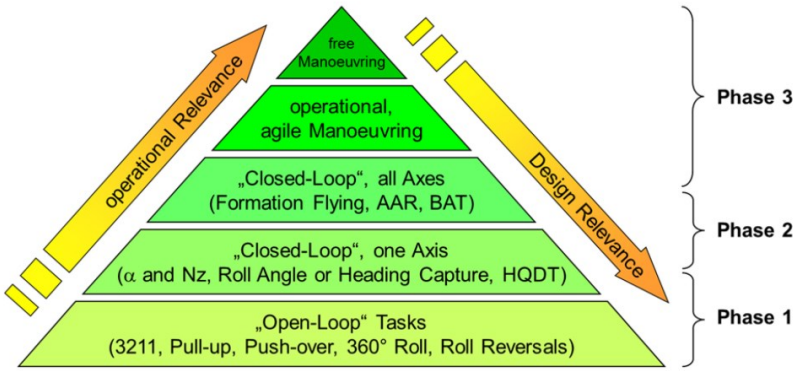


Abbildung 3-8: Strukturansatz für den Flugerprobungsprozess [32]

Im Folgenden werden die für diese Arbeit relevanten Manöver Take-Off, Steig- und Sinkflug, Stationärer Kurvenflug, Dutch Roll, Elevator Sweep, Steady Pull-Up und Steady Push-Over sowie Descent, Approach, Landing beschrieben. Im Vordergrund stehen hier vor allem Manöver zur Ermittlung des Neutral- und Manöverpunkts, der Steig- und Sinkfähigkeit im Rahmen der Flugzeugpolare sowie Manöver zur Untersuchung der natürlichen Eigenfrequenz des Flugzeugrumpfes bzw. der Tragflächen. Die Bedeutung dieser Aspekte für die Stabilität und Steuerung eines Luftfahrzeugs wurden in Kapitel 3.1 bereits näher beschrieben. Einige

gängige Manöver werden in diesem Fall außen vorgelassen, die ebenfalls eine wichtige Rolle bei der Flugeigenschaftsvermessung spielen. Allerdings sind diese nicht Teil der getroffenen Manöverauswahl und werden daher für die vorliegende Arbeit nicht weiter berücksichtigt.

Take-Off

Der Take-Off (dt.: Abflug) erstreckt sich in der Regel von der Startposition auf der Start- und Landebahn bis zur erreichten Flughöhe über Grund von 1500 ft (457,2 m). Das gesamte Manöver kann dabei in unterschiedliche Phasen unterteilt werden. Der Start zeichnet sich dadurch aus, dass anfangs die im Flugplan vorgesehene Startprozedur inklusive der Einstellung des Schubs, der Klappen und der Spoiler durchlaufen wird. Nach dem Lösen der Standbremse wird das Flugzeug auf eine vorgegebene Geschwindigkeit beschleunigt, woraufhin eine Zugkraft am Steuerhorn aufgebracht wird, um die Flugzeugnase hochzuziehen. Anschließend muss das Flugzeug bis zu einer Flughöhe von 35 ft (10,668 m) auf eine weitere Geschwindigkeit V_2 beschleunigt werden. V_2 darf von dort an bis zu einer Flughöhe von 400 ft (121,92 m) nicht mehr unterschritten werden. Wenn im Anschluss die Steigrate stabilisiert ist, wird das Fahrwerk eingefahren. In der anschließenden Steigflugphase wird die Flughöhe wie beschrieben auf 400 ft erhöht, wo das Flugzeug optional ausgetrimmt und die finale Abfluggeschwindigkeit V_{FTO} eingestellt werden kann. Schließlich wird unter V_{FTO} die angestrebte Reiseflughöhe von 1500 ft oder höher angeflogen. Ergänzend dazu existieren verschiedene Methoden zur Durchführung von Take-Offs, die in dieser Arbeit nicht weiter berücksichtigt werden. Abbildung 3-9 und Abbildung 3-10 zeigen den schematischen Ablauf eines Abflugs sowie beispielhaft den entsprechenden Datenstrom der gemessenen Flughöhe aus den Flugversuchen. [34]

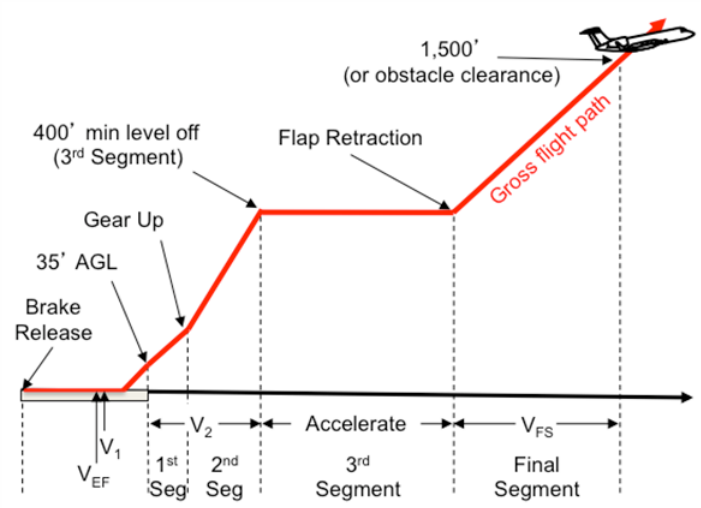


Abbildung 3-9: Prozedur Take-Off [35]

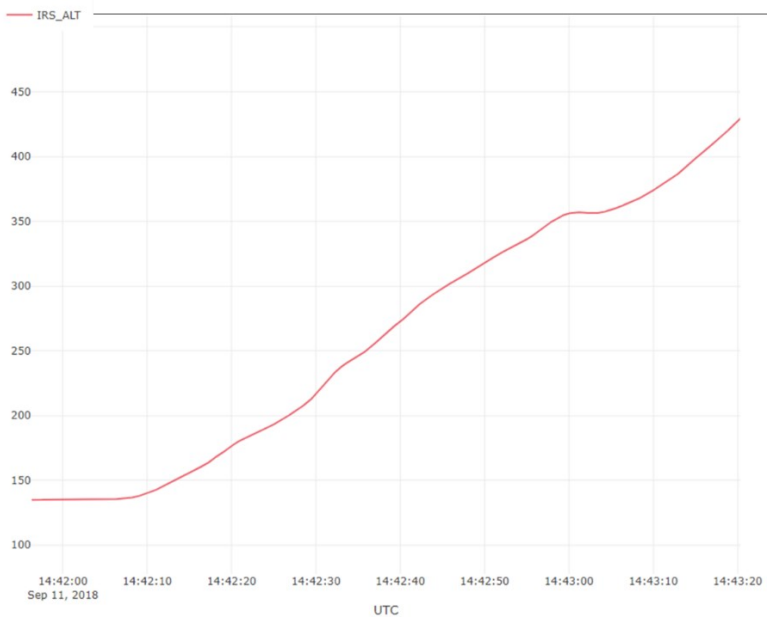


Abbildung 3-10: Messignal Flughöhenprofil (IRS_ALT) Take-Off

Dutch Roll

Das Dutch Roll Manöver (auch: Dutch Roll Mode oder Taumelschwingung) ist eine der charakteristischen dynamischen Seitenbewegungen eines Flugzeugs, die während des stabilen Fluges unterdrückt werden. Die Bewegung kann durch eine zusammengesetzte Schwingung bestehend aus Gieren, Schiebeflug und Rollen beschrieben werden (Abbildung 3-11). Im Normalflug wird diese Bewegung entweder versehentlich durch den Piloten oder durch äußere Einflüsse hervorgerufen. Dieses Manöver wird im Testflug verwendet, um z. B. die Bewegungsfrequenz, das Dämpfungsverhalten während des Manövers sowie den Einfluss der Gier- auf die Rollrate anhand der Verhältnisse zwischen Roll- und Gierwinkel zu untersuchen. Dabei erfolgen im stabilen, ausgetrimmten Zustand bei einer definierten Geschwindigkeit alternierende Seitenrudereingaben nach links und rechts. Somit wird die Dutch Roll Bewegung angeregt und verstärkt. Dieser Prozess wird solange wiederholt bis die angestrebte Schwingungsamplitude erreicht wurde. Anschließend wird die Seitenruderstellung wieder in den ausgetrimmten Zustand versetzt. Die dadurch entstehende Roll-Gier-Schwingung weist eine Phasenverschiebung von 90°-180° auf und wird mit verschiedenen Geschwindigkeiten sowie ein- oder ausgeschaltetem Gierdämpfer geflogen. Beispielhaft zeigt Abbildung 3-11 die Prozedur eines Dutch Roll Manövers und Abbildung 3-12 eine entsprechende Sequenz aus realen Messdaten. Darin sind die zeitlichen Verläufe der Roll- und Gierrate überlagert dargestellt. [36]

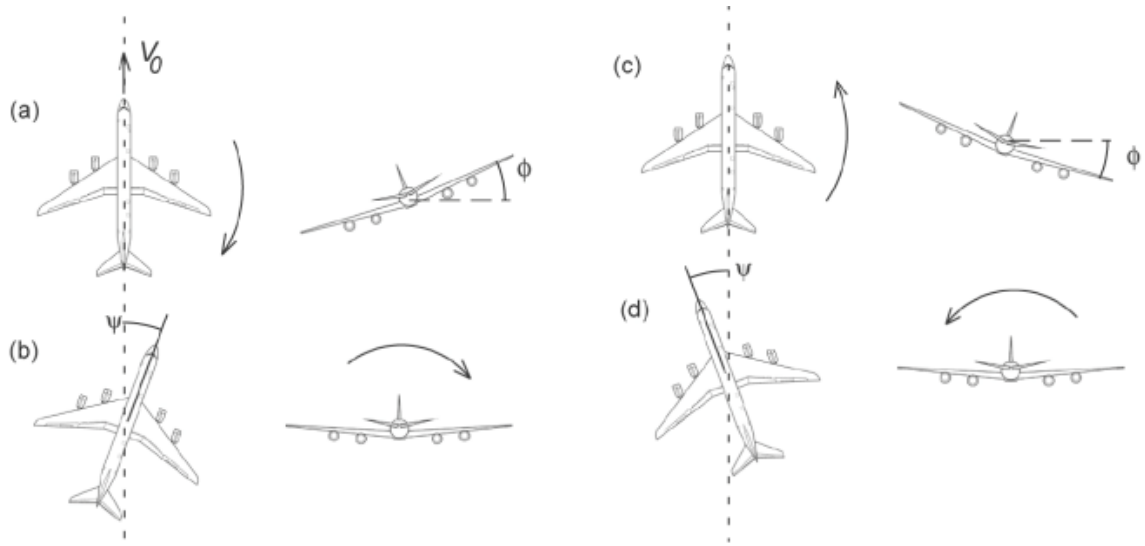


Abbildung 3-11: Dutch Roll Bewegungsablauf [37]

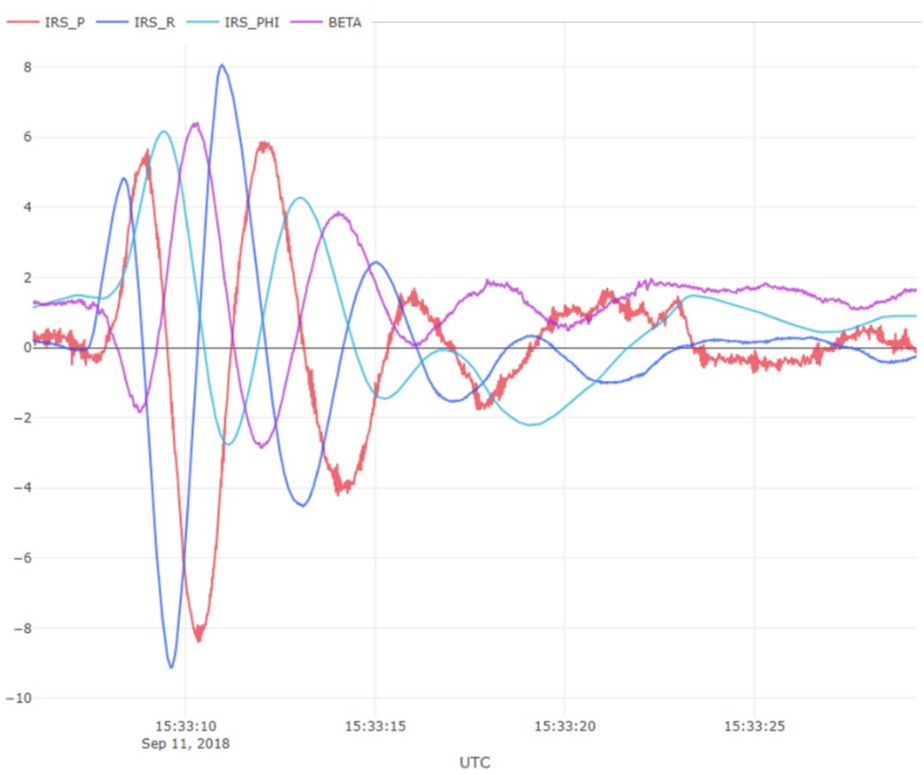


Abbildung 3-12: Messsignale p (IRS_P), r (IRS_R), Φ (IRS_PHI) und β (BETA) Dutch Roll Manöver

Elevator Sweep

Da aeroelastische Phänomene sicherheitsrelevanten Einfluss auf den Flugzustand nehmen, ist die Eigenfrequenz von Flugzeugrumpf und Tragflächen von erhöhtem Interesse für die Flugzeugentwicklung. Daher ist u. a. die Eigenfrequenz eine notwendige Angabe zur Zulassung eines Flugzeugs. Das Elevator Sweep Manöver dient der Ermittlung dieser

Eigenfrequenz. Zunächst wird dafür aus dem ausgetrimmten Flugzustand bei einer definierten Geschwindigkeit eine niederfrequente Sinus-Höhenschwingung eingeleitet. Anschließend wird bei konstanter Amplitude die Schwingungsfrequenz sukzessive bis zur Eigenfrequenz und darüber hinaus gesteigert, wie in Abbildung 3-13 und Abbildung 3-14 erkennbar ist. Der Indikator für die Resonanz ist die Reaktion des Flugzeugs auf die Steuereingabe mit einem Phasenversatz von 90° (Abbildung 3-15). [32]

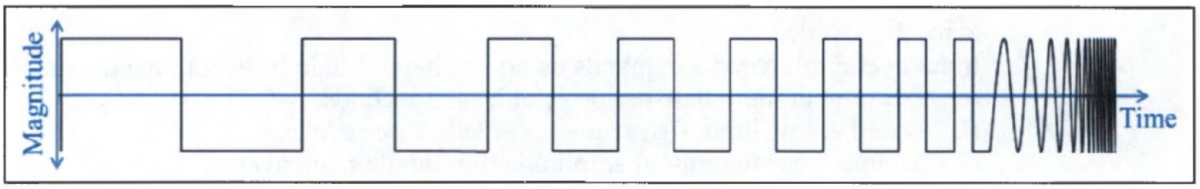


Abbildung 3-13: Schematischer Amplitudenverlauf des Höhenruders beim Elevator Sweep [32]

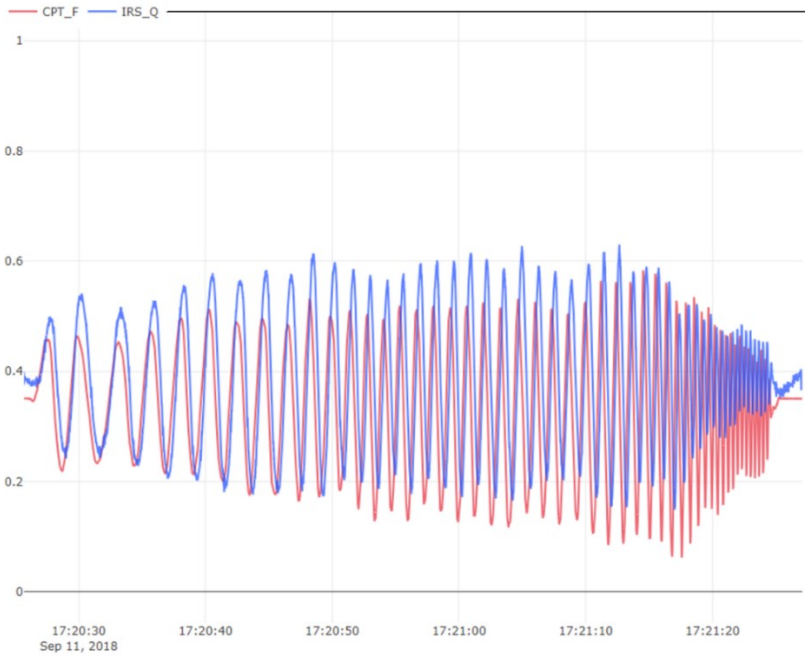


Abbildung 3-14: Messignale Höhenruderwinkel η (CPT_F) und q (IRS_Q) Elevator Sweep Manöver

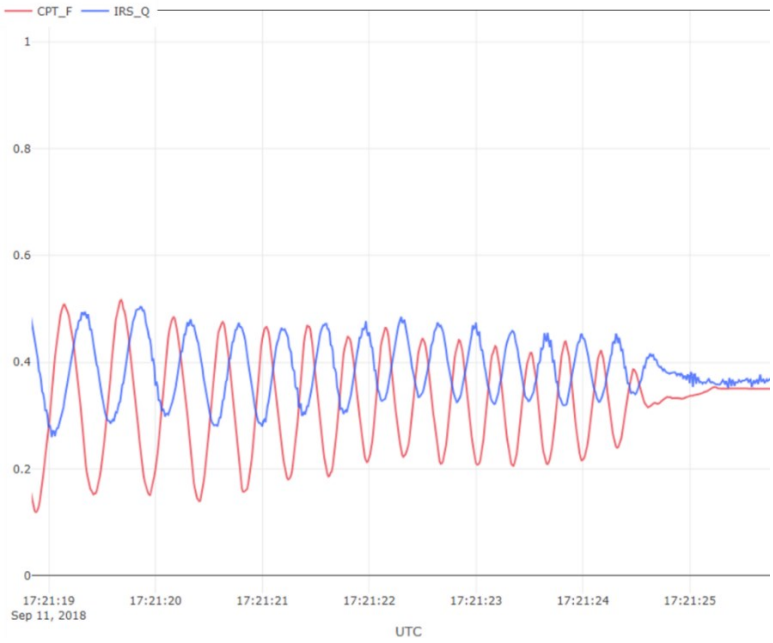


Abbildung 3-15: 90° Phasenversatz von η und q Elevator Sweep Manöver

Steady Pull Up/Steady Push Over

Um den Manöverpunkt und damit die Manöverstabilität eines Flugzeugs zu ermitteln werden unter anderem Abfangbögen durchflogen (Kap 3.1.2). Dies erfolgt im Rahmen von einem Steady Pull-Up oder Steady Push-Over Manöver. Da das Steady Push-Over Manöver auch als ein „umgekehrtes Steady Pull-Up“-Manöver [36] bezeichnet wird, beschreibt dieser Abschnitt stellvertretend das Steady Pull-Up-Manöver im Detail. Dabei wird ausgehend vom

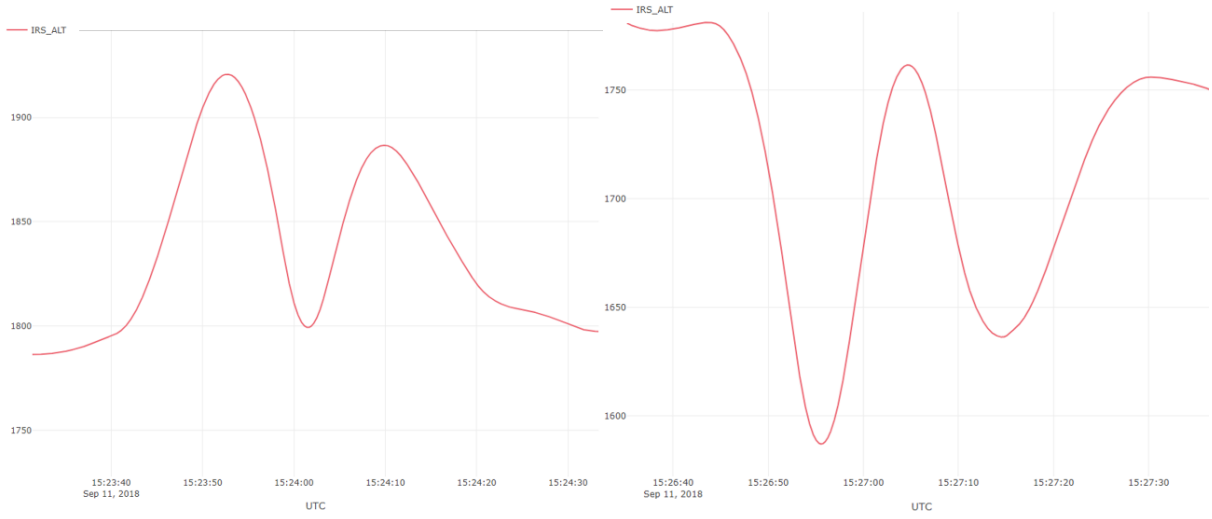


Abbildung 3-16: Flughöhenprofile (IRS_ALT) Steady Pull-Up (links) und Steady Push-Over Manöver (rechts)

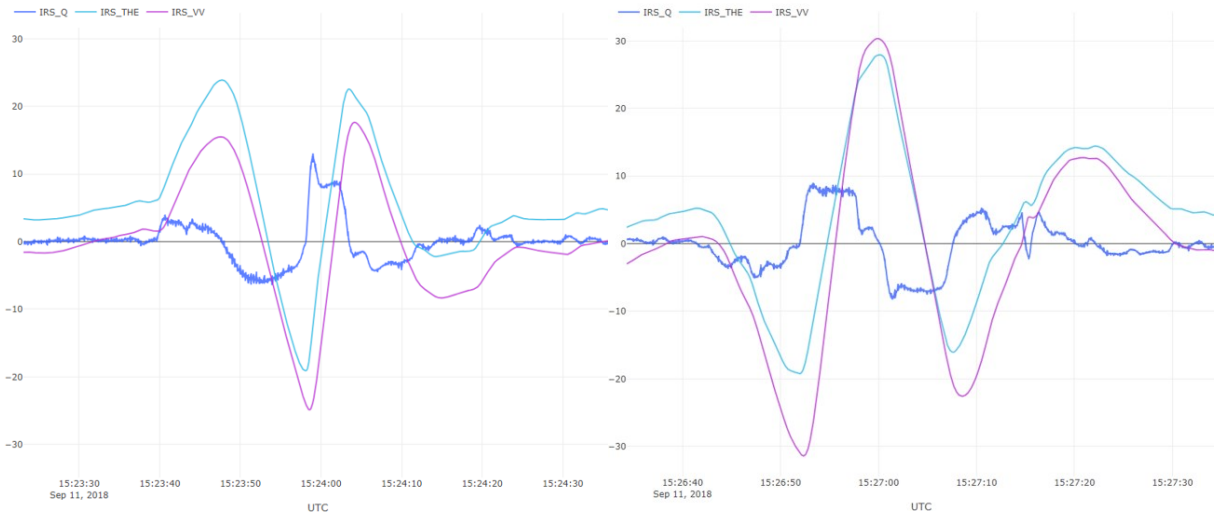


Abbildung 3-17: Messignale q (blau, IRS_Q), θ (türkis, IRS_THE), Steiggeschwindigkeit (violet, IRS_VV) Steady Pull-Up (links) und Steady Push-Over Manöver (rechts)

ausgetrimmten Flugzustand bei definierter Ausgangsgeschwindigkeit, -flughöhe und -schub ein Steigflug eingeleitet. Nachdem dadurch das Flugzeug auf eine vorab festgelegte Fluggeschwindigkeit verzögert wird, drückt der Pilot durch eine negative Nickbewegung die Flugzeugnase nach unten, wodurch sich Geschwindigkeit und Flughöhe wieder den Ausgangswerten annähern. Beim Erreichen der Ausgangshöhe, wendet der Pilot eine Zugkraft am Steuerorgan auf, um die Nase des Flugzeugs erneut nach oben zu ziehen und einen Abfangbogen zu durchfliegen. Innerhalb des kurzen Zeitabschnitts, in dem der Nickwinkel θ bei $\theta \cong 0$ liegt, werden die aufgewandte Zugkraft, die Höhenruderposition und die Normalkraft abgelesen. Im Anschluss wird nach erneuter Verzögerung des Flugzeugs im Steigflug die Ausgangsflughöhe angesteuert und das Manöver beendet. Das Steady Push-Over Manöver verläuft genau umgekehrt, sodass zunächst die Geschwindigkeit durch einen kurzen Sinkflug gesteigert wird. Anschließend werden während eines Steigfluges Fluggeschwindigkeit, Flughöhe und Schub wieder den Ausgangswerten angeglichen. Daraufhin wird ein Abfangbogen durchflogen, indem der Pilot eine Druckkraft am Steuerhorn aufwendet. Analog zum Steady Pull-Up Manöver wird das Manöver anschließend durch das Ansteuern der Ausgangsflughöhe beendet. Zur besseren Verständlichkeit der Prozeduren sind in Abbildung 3-16 die Höhenprofile beider Manöver als Messignale dargestellt. Das Steady Push-Over Manöver gilt als optimale Methode zur Untersuchung der Manöverstabilität eines Flugzeugs im Lastbereich von unter 1 g. Während der Durchführung der Manöver, sollten die Wertetoleranzen von Fluggeschwindigkeit, Flughöhe und Anstellwinkel verglichen zum Ausgangsflugzustand eingehalten werden. Für die Fluggeschwindigkeit $KIAS_t$ gilt dabei $KIAS_t = KIAS_{Start} \pm 5\,KIAS$. Für die Flughöhe h_t gilt $h_t = h_{Start} \pm 2000\,ft$. Für den

Anstellwinkel α_t gilt $\alpha_t = \alpha_{Start} \pm 15°$. Abbildung 3-17 zeigt dazu die manövertypischen Signalverläufe der Nickgeschwindigkeit q, des Nickwinkels θ und der Steiggeschwindigkeit. [36]

Kurvenflug

Kurvenflüge werden zum einen als allgemeine Orientierungsänderung und zum anderen als Manöver zur Ermittlung von Flugzeugeigenschaften in verschiedenen Ausführungen geflogen. Im Rahmen dieser Arbeit werden lediglich Kurvenflüge zur Orientierungsänderung kurz erläutert. Diese werden mithilfe der Querruder durch eine Rollbewegung und die Betätigung des Seitenruders in Kurvenrichtung eingeleitet. Während des Kurvenfluges wird die Flughöhe je nach Vorgabe und Ziel maßgeblich durch Höhenrudereingaben entweder gehalten, erhöht oder verringert. Abbildung 3-18 zeigt ergänzend dazu Beispieldaten der Flugzeugorientierung anhand des Gierwinkels ψ (auch: Heading) sowie des Rollwinkels Φ einer geflogenen Linkskurve.

Zur Ermittlung des Manöverpunktes und damit auch der Manöverstabilität werden zudem Variationen des sog. stationären Kurvenflugs durchgeführt. Dabei wird eine Kurve bei konstanter Flughöhe und Fluggeschwindigkeit mit variierenden Hängewinkeln und Lastfaktoren geflogen. In der Querlage wird dabei das Scheinlot, also der resultierende Vektor aus Gewichts- und Zentrifugalkraft, parallel zur Flugzeughochachse mithilfe der Seitenrudersteuerung eingestellt. [30]

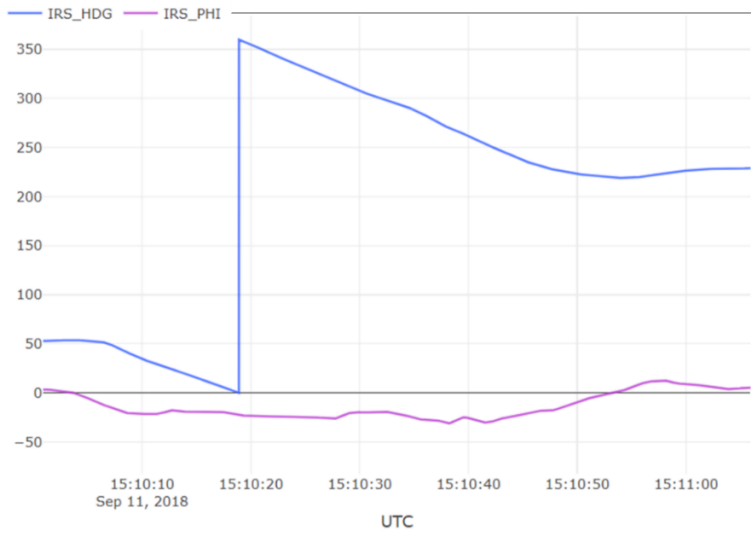


Abbildung 3-18: Messignale ψ (IRS_HDG) und Φ (IRS_PHI) Linkskurvenflug

Steig- und Sinkflug (Saw Tooth)

Die Steigleistung ist ein wichtiger Indikator für Steiggeschwindigkeit, Steigzeit, Flugzeit und Kraftstoffverbrauch eines Flugzeugs und ist zudem sicherheitsrelevant. So muss beispielsweise die verbleibende Flugleistung bei Ausfall eines von mehreren Triebwerken für eine sichere Beendigung des Fluges ausreichen [1]. Das Flugzeugverhalten bei Steig- und Sinkflügen (engl.: climb und descent) ist maßgeblich für die Steigleistung und wird daher im Flugversuch gezielt untersucht. Dabei werden außerdem wichtige Erkenntnisse über die Ausprägung der Flugzeugpolare gewonnen. Die Durchführung dieses Teils eines Flugversuchs wurde unter anderem von Raab und Burwitz im Flugversuchsprogramm der „DLR Summer School“ detailliert [30]. Bei der Durchführung wird demnach eine mittlere Höhe h_m vorgegeben, um welche herum abwechselnd Steig- und Sinkflüge mit unterschiedlichen Geschwindigkeiten geflogen werden. Daraus resultiert ein sägezahnähnliches Flughöhenprofil, weshalb die alternierende Durchführung von Steig- und Sinkflügen auch als Saw Tooth (dt. Sägezahn) bezeichnet wird [1]. Da die Motorleistung je Steig- oder Sinkflug konstant gehalten werden soll, darf laut Raab und Burwitz die Höhenabweichung um eine mittlere Flughöhe h_m nicht zu hoch sein [30]. Vor Beginn des Manövers wird die Leistung ca. eine Minute lang stabilisiert [1]. Später wird im Versuch für den Sinkflug die Leistungseinstellung stark reduziert allerdings nie komplett null. Weiterhin sollen bei der Durchführung möglichst Bedingungen der Normalatmosphäre nach ICAO-Standardatmosphäre herrschen. Darüber hinaus werden Steig- und Sinkflüge im Verlauf eines Flugversuches immer wieder notwendig, da für die Durchführung verschiedener Manöver meist unterschiedliche Flughöhen als Rahmenbedingung vorgesehen werden. Abbildung 3-19 veranschaulicht den Ablauf des oben beschriebenen Manövers anhand des charakteristischen Flughöhenprofils.

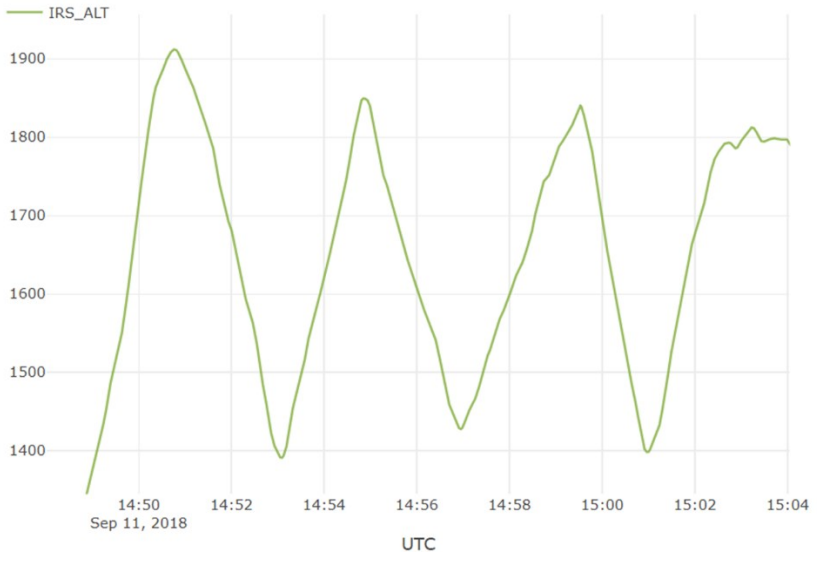


Abbildung 3-19: Flughöhenprofil (IRS_ALT) Saw Tooth Manöver

Descent, Approach, Landing

Im Flugversuch muss das Handling des Flugzeugs während des Landeanflugs und der Landung selbst erprobt werden. Dazu werden Landeanflug und Landung mithilfe verschiedener Methoden mit ebenfalls unterschiedlichen Flugzeugkonfigurationen durchgeführt. Grundlegend wird der Landeanflug mit einem Sinkflug ausgehend von der Reiseflughöhe eingeleitet. Daraufhin folgt der Endanflug (engl.: approach), wobei das Flugzeug passend zur Orientierung der Landebahn ausgerichtet, die Flughöhe weiterhin langsam verringert und das Fahrwerk ausgefahren wird. Bei der allgemeinen Landeprozedur folgt daraufhin die Landung (engl.: landing), welche durch das Aufsetzen des Fahrwerks auf der Landebahn startet. Daraufhin wird das Flugzeug unter Verwendung verschiedener Methoden auf der Landebahn bis zum Stillstand abgebremst. So wird z. B. die schnellste und effizienteste Methode zur Landung unter Einhaltung der Belastungstoleranzen von Fahrwerk und Bremsen erprobt. Auch hier gibt es zahlreiche Methoden zur Durchführung einer Landung, die für diese Arbeit jedoch nicht detaillierter betrachtet werden. [32]

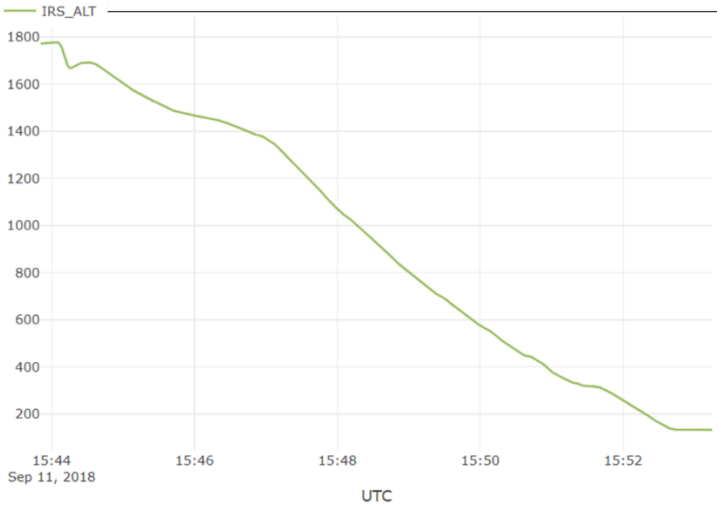


Abbildung 3-20: Höhenprofil (IRS_ALT) Descent, Approach, Landing

3.3 Deep Learning

Zur Spezifizierung des Begriffs der Deep Neural Networks wird der Begriff des Deep Learning und dessen Verhältnis zum Machine Learning (ML, dt.: Maschinelles Lernen) zunächst kurz erläutert. DL ist als Teilmenge der übergeordneten Gruppe von ML-Algorithmen zu verstehen [38]. Konventionelle ML-Algorithmen wurden in der Vergangenheit und werden heute noch für verschiedene Aufgaben zur Automatisierung von Prozessen verwendet [39] [14]. Je komplexer diese Aufgaben allerdings werden, desto mehr stoßen die klassischen ML-Methoden an ihre Grenzen [11]. Grund dafür ist, dass klassische Ansätze in vielerlei Hinsicht meist von einem

tiefen Verständnis und Expertenwissen über das jeweilige Fachgebiet abhängig sind. Die Schwäche dieser Ansätze zeigt sich laut LeCun darin, dass sie aus Rohdatensätzen Merkmale und Metadaten nur schwer identifizieren und in erforderlicher Qualität darstellen können [11]. An dieser Stelle setzt demnach das Prinzip des Representation Learning (dt. Repräsentatives Lernen) an. Es beschreibt die geeignete Darstellung von Informationen aus Rohdatensätzen ohne aufwändige Vor- oder Nachbearbeitung, um anhand dessen u. a. direkte Klassifizierungen oder Prognosen durchführen zu können. Im DL wird diese Idee aufgegriffen und skaliert, indem Informationen aus den Rohdaten extrahiert und auf mehreren tiefergreifenden Ebenen immer stärker transformiert und abstrahiert werden [11] [38]. Somit können u. a. präzisere Vorhersagen oder Klassifizierungen für komplexere und größere Datenmengen getroffen werden. Dafür bedienen sich DL-Methoden der Struktur von Artificial Neural Networks (ANN, dt.: Künstliche Neuronale Netze), welche dem menschlichen Gehirn und dessen Lernprozessen nachempfunden sind.

3.3.1 Künstliche Neuronale Netze

Mittlerweile existieren zahlreiche Netzarchitekturen, die verschiedenen Zwecken und Funktionen dienen. Zur Erklärung der allgemeinen Funktionsweise von ANNs dient im Folgenden die vergleichsweise einfache Architektur des MLP als stellvertretendes Beispiel. Darin sind mehrere Knotenpunkte (Neuronen und ein Bias-Neuron) in verschiedenen Schichten (engl.: layer) angeordnet. Jedes einzelne Neuron einer Schicht wird durch gewichtete Verbindungen mit jedem Neuron der nachfolgenden Neuronenschicht vernetzt. Dies wird als sogenanntes Fully Connected Layer (dt.: vollvernetzte Schicht) bezeichnet. So entsteht eine Netzstruktur wie in Abbildung 3-21 abgebildet. Im Fall der tiefen Neuronalen Netze ist ein Netz immer mit mindestens drei verschiedenen Arten von Schichten ausgestattet: die Eingabeschicht (engl.: input layer), die verdeckte Zwischenschicht (engl.: hidden layer) und die Ausgabeschicht (engl.: output layer). Dabei ist von einem DNN erst die Rede, wenn ein Netz mit mindestens zwei verdeckten Schichten - also insgesamt mindestens vier Schichten - ausgestattet ist [40].

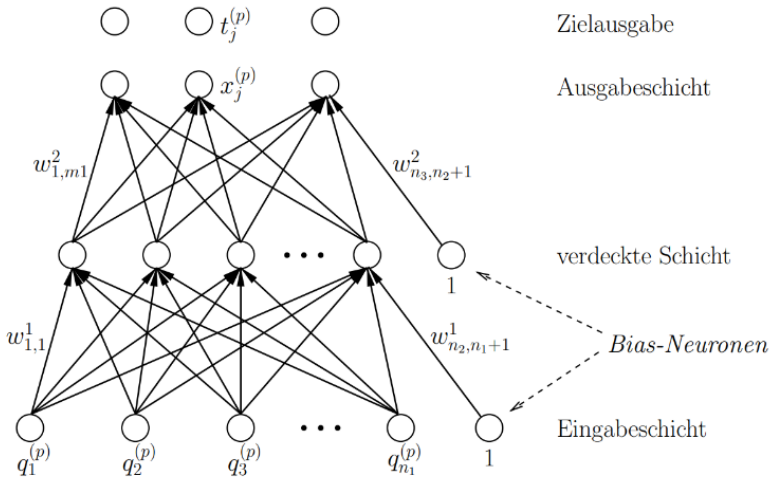


Abbildung 3-21: Dreilagiges Backpropagation Netz [41]

In einem ANN werden die Merkmale eines Datensatzes als n-dimensionaler Zahlenvektor (in Abbildung 3-19 mit q gekennzeichnet) in das Netz eingespeist. Ertel und Black führen dazu aus, dass die Neuronenwerte x_i einer beliebigen Schicht i Auskunft darüber geben, wie stark ein Neuron jeweils aktiviert ist [41]. Bei der Betrachtung zweier aufeinanderfolgender Neuronenschichten i und j wird die Aktivierung der Neuronen x_i in der ersten Schicht in einem Eingangsvektor zusammengefasst. Alle Eingangsvektorelemente werden mit den gewichteten Verbindungen w_{ij} multipliziert, die zu den verbundenen Neuronen x_j führen. Die daraus resultierenden Skalare werden aufsummiert und an die Aktivierungsfunktion $f(x)$ zur Aktivierung des Neurons der nachfolgenden Schicht j übergeben. Die Summe der gewichteten Eingangswerte dient hierfür als Funktionswert x, welcher in $f(x)$ eingesetzt wird. Daraus resultiert die Formel zur Berechnung der Aktivierung eines Neurons in der Schicht j:

$$x_j = f\left(\sum_{i=1}^{n} w_{ij} x_i\right) \qquad 3.2$$

Aktivierungsfunktionen

Als Aktivierungsfunktion können verschiedene mathematische Funktionen verwendet werden. Das ermöglicht es dem Benutzer ebenfalls nicht lineare Funktionen zu diesem Zweck zu verwenden. Somit können durch ANNs und speziell durch DNNs komplexere Datenzusammenhänge und -muster erkannt werden. Mit am häufigsten dienen dafür die Sigmoid- (Gl. 3.3, Abbildung 3-22 (a)), die Tangens-Hyperbolicus- (Gl. 3.4, Abbildung 3-22 (c)), die Rectified Linear Unit- (Gl. 3.5, Abbildung 3-22 (b)) sowie die Softmax-Funktion (Gl. 3.6) [42]:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.3}$$

$$f(x) = \tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{3.4}$$

$$f(x) = \max(0, x) \tag{3.5}$$

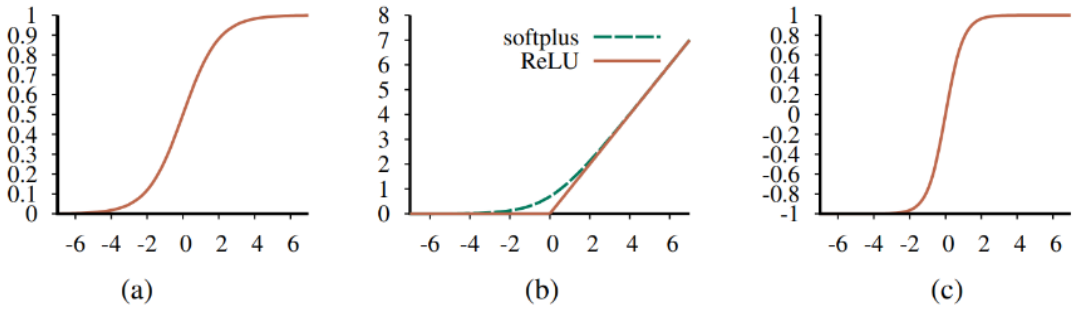


Abbildung 3-22: Aktivierungsfunktionen Sigmoid (a), ReLU (b) und tanh (c) [42]

Die Softmax-Funktion stellt eine weitere Aktivierungsfunktion dar, welche hauptsächlich bei Klassifikatoren mehrerer Klassen verwendet wird. Für eine Klassifizierung mit d möglichen Klassen wird mithilfe der Softmax-Funktion ein Vektor mit d Wahrscheinlichkeitswerten, welche aufsummiert den Wert 1 ergeben, aus einem ANN ausgegeben. Der mathematische Ausdruck für die Aktivierung eines Neurons lautet dafür wie folgt:

$$f(x) = \frac{e^{x}}{\sum_{k=1}^{d} e^{x_k}} \tag{3.6}$$

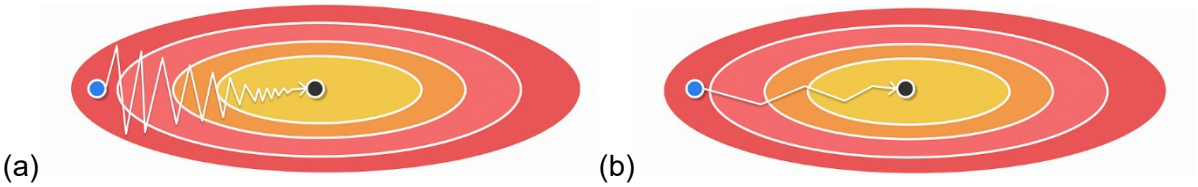
Backpropagation mit Fehlerfunktion und Optimierungsalgorithmus

Da sowohl Eingabevektor als auch Aktivierungsfunktionen invariant sind, zielt das Training eines ANN auf die Anpassung von den Gewichten der Neuronenverbindungen ab [42]. In diesem Zusammenhang nimmt der Backpropagation-Algorithmus eine wichtige Rolle im Bereich des Trainings tiefer neuronaler Netze ein [42]. Dafür wird zunächst anhand der Abweichung des Ausgabewerts vom erwarteten Zielwert eine Fehlerfunktion L (engl.: Loss oder Loss Function) definiert, die abhängig von den anzupassenden Gewichten W ist [42]. Dieser Prozess des Lernens mit gelabelten Datensätzen wird auch als Supervised Learning (dt.: überwachtes Lernen) bezeichnet [42]. Je nach Problemstellung werden dazu unterschiedliche Fehlerfunktionen verwendet. Für ein Klassifizierungsproblem ist beispielsweise der sogenannte Cross-Entropy Loss (dt.: Kreuzentropie Fehlerfunktion) zu nennen (Gl. 3.7) [43]. Als Klassifikator geben ANNs meist Wahrscheinlichkeiten p_c pro Klasse c aus, die als Zuversicht der Vorhersage eines Netzes interpretiert werden können [43]. Cross-

Entropy ist für diesen Fall ausgelegt und wächst speziell bei zuversichtlichen Klassifizierungen, die nicht der wahren Klasse y_c entsprechen, stärker an [43]:

$$L(W) = -\sum_{c=1}^{N} y_c * \log(p_c) \tag{3.7}$$

Ziel des Trainings eines ANNs ist es nach Russel und Norvig das globale Minimum dieser Fehlerfunktion zu finden [42]. Dafür werden unter Verwendung der Kettenregel die partiellen Ableitungen der Fehlerfunktion L nach den Gewichten des Netzes W, die als Variablen für die Vorhersagen p_c dienen, berechnet. Daraus resultiert ein Gradient, der von Schicht zu Schicht eines ANN auf alle Gewichte angewandt wird. Diese Operation wird solange wiederholt bis die Gewichte sich nicht mehr um einen bestimmten Wert verändern oder bis eine definierte maximale Trainingsdauer erreicht ist. Dieser Prozess wird als Stochastic Gradient Descent (SGD, dt.: stochastischer Gradientenabstieg) bezeichnet (Abbildung 3-23 (a)). Um dabei das globale Minimum schneller und überhaupt zu erreichen, können verschiedene Algorithmen angewandt werden, welche die Suche optimieren. Ein prominentes Beispiel hierfür ist der Optimierungsalgorithmus Adaptive Moment Estimation (ADAM, dt.: adaptive Momentschätzung) [44]. Dieser erweitert die Methode des SGD, sodass erstens lokale Minima und Sattelpunkte in der Hyperebene der Verlustfunktion besser überwunden werden können und zugleich das Training bei schwachen Gradienten beschleunigt wird. Zweitens wird der Weg zum globalen Minimum der Verlustfunktion verkürzt, sodass das Risiko in lokalen Minima oder Sattelpunkten zu konvergieren, sinkt (Abbildung 3-23 (b)).



(a) (b)

Abbildung 3-23: Vergleich Optimierung der Gewichte mit SGD (a) und ADAM (b) [45]

Dazu wird eine Abhängigkeit des aktuellsten Gradienten von den vorherigen Gradienten aufgebaut (Momentum), wodurch ADAM die Lernrate während des Trainings anpasst.

Metriken für Klassifizierungsalgorithmen

Um zu entscheiden, ob ein ANN eine vorgesehene Aufgabe gut oder schlecht bewältigt, muss die Leistungsfähigkeit des ANN evaluiert werden. Dies kann anhand verschiedener Qualitätsmaße ermittelt werden, wobei die Wahl der Maße zur Bewertung vom Anwendungsfall abhängt. Für ein Klassifizierungsproblem beziehen sich die gängigsten Qualitätsmaße auf die Mengen positiver und negativer Klassenzugehörigkeiten von Daten, die

durch ein ANN klassifiziert wurden. Die positiven Daten werden unterteilt in True Positives (TP, dt.: wahr positive) und False Negatives (FN, dt.: falsch negative). Als TP gelten die Daten, die einer gelabelten Klasse korrekterweise zugeordnet wurden. Als FN gelten dagegen die Daten, die einer Klasse fälschlicherweise nicht zugeordnet wurden. Analog werden die negativen Daten in True Negatives (TN, dt.: wahr negative) und False Positives (FP, dt.: falsch positive) unterteilt. Eines der aussagekräftigsten Qualitätsmaße für einen Klassifikator ist der sogenannte Recall (auch: Sensitivität), der beschreibt, welcher Anteil einer gelabelten Klasse durch den Algorithmus tatsächlich erkannt wurde. [41]

Der Recall berechnet sich wie folgt:

$$Recall = \frac{TP}{TP + FN} \tag{3.8}$$

Im Gegensatz dazu steht die sogenannte Precision (auch: positiver Vorhersagewert). Diese beschreibt den Anteil der korrekt vorhergesagten positiven Klassenlabels innerhalb aller positiven Vorhersagen eines Algorithmus [41]. Die Precision berechnet sich wie folgt:

$$Precision = \frac{TP}{TP + FP} \tag{3.9}$$

Sollen beide Kriterien von einem Algorithmus möglichst erfüllt werden, wird der sogenannte F- oder F1-Score (auch: F-Maß) verwendet [41]. Dieser beschreibt das harmonische Mittel von Recall und Precision:

$$F1 = \frac{2 * Recall * Precision}{Recall + Precision} \tag{3.10}$$

Speziell für die Bewertung von Klassifizierungs-Algorithmen, die auf unausgeglichenen Trainingsdaten trainiert wurden, ist der Matthews Correlation Coefficient (MCC) ein weiteres aussagekräftiges Qualitätsmaß [46]. Dieser trifft mit einem Wert zwischen -1 und 1 eine Aussage über die Korrelation zwischen vorhergesagter und wahrer Klasse [46]. Der Wert 1 steht dabei dafür, dass alle Vorhersagen eines Klassifikators dem Erwartungswert entsprechen [46]. Im Gegensatz dazu steht der Wert -1 für eine stark negative Abhängigkeit, sodass die Vorhersage eines Klassifikators genau dem gegenteiligen Wert des Erwartungswertes entspricht [46].

Eine große Herausforderung von Tiefen Neuronalen Netzen besteht zurzeit darin, dass es sich noch um eine Black Box handelt, d. h. es ist aktuell noch schwer bis kaum vermittelbar, wieso ein DNN zu einer bestimmten Vorhersage oder Klassifizierung kommt [47]. Das fehlende Verständnis mindert mitunter die Akzeptanz für den Einsatz von Neuronalen Netzen [47]. Die einfache Skalierbarkeit und die Fähigkeit, hochkomplexe nichtlineare Modelle mithilfe von

DNNs abbilden zu können, machen Sie allerdings zu einem wichtigen Werkzeug mit hohem Potenzial immer komplexere Aufgaben lösen zu können [11].

3.3.2 Bias-Variance Tradeoff

Zwei häufig auftretende Probleme beim Training Neuronaler Netze sind die Verzerrung (engl.: bias) und die Varianz (engl.: variance) [48]. Zudem ist die Verzerrung eng mit dem Begriff der Unteranpassung (engl.: Underfitting) und die Varianz eng mit dem Begriff der Überanpassung (engl.: Overfitting) verbunden [48]. Underfitting bedeutet daher, dass ein Modell eine hohe Verzerrung und geringe Varianz aufweist, wohingegen Overfitting impliziert, dass ein Modell eine geringe Verzerrung und eine hohe Varianz aufweist [48]. Unterangepasste Modelle weisen eine zu geringe Komplexität auf, sodass sie sich nicht auf komplexe Datenmuster in Trainingsdaten anpassen und diese ausreichend genau erkennen können. Dagegen ist Overfitting weithin bekannt als starke Differenz zwischen der Leistungsfähigkeit eines Modells beim Training und der Leistungsfähigkeit bei der Modellevaluierung auf unbekannten Daten [49]. Ursächlich dafür können z. B. laut Lang eine hohe Modellkomplexität, zu kleine Trainingsdatensätze, schlecht ausgewählte Datenmuster oder zu viele durchgeführte Trainingsepochen sein. Beim Versuch die Verzerrung eines Modells zu verringern, erhöht sich jedoch meist die Varianz und ebenso umgekehrt, sodass bei der Optimierung hin zu einem ausgeglichenen Modell ein Kompromiss zwischen beiden Kriterien gefunden werden muss [50]. Dieser ist auch unter dem Begriff des Bias-Variance Tradeoff (dt.: Verzerrung-Varianz-Kompromiss) bekannt [50]. Dazu können verschiedene Methoden angewandt werden, um Under- oder Overfitting entgegenzuwirken, wovon einige in Tabelle 3-2 aufgelistet werden.

Die Dimensionalität der Eingangsdaten und die Modellkomplexität entscheiden in diesem Zusammenhang mit über die Komplexität der Merkmale, die durch ein Modell in den Daten erkannt werden können [48]. Diesbezüglich kann je nach Problemstellung die Wahl der Modellarchitektur ebenfalls entscheidend sein [48]. Early Stopping überwacht den Wert eines definierten Scores und stoppt das Training frühzeitig, falls dieser Score sich innerhalb einer definierten Epochenanzahl nicht mehr um eine bestimmte Differenz verbessert [51]. Mithilfe von angewandten Regularisierungstechniken werden zu komplexe Modelle zusätzlich bestraft, um Overfitting zu vermeiden.

Tabelle 3-2: Methoden zur Verringerung von Under- und Overfitting

Underfitting/Verzerrung	Overfitting/Varianz
Anzahl relevanter Parameter erhöhen [52]	Dimensionalität der Daten reduzieren (Parameteranzahl) [49]
Komplexere Modellauslegung [48]	Modellkomplexität verringern [48]
Regularisierungstechniken reduzieren [52]	Regularisierungstechniken anwenden [48]
Modellarchitektur ändern [48]	Trainingsdatensatz erweitern, Variabilität in den Daten erhöhen (Datenaugmentierung) [48]
Trainingsdauer erhöhen [48]	Modellarchitektur ändern [48]
Kombination verschiedener Modelle (Ensembling) [52]	Frühzeitiges Stoppen des Trainings-prozesses (Early Stopping, dt.: frühes Stoppen) [51]

In Bezug auf Neuronale Netzarchitekturen ist in diesem Zusammenhang die Verwendung von Dropout-Schichten zu nennen, die von Srivastava et al. erforscht wurden [53]. Dropout-Schichten sorgen demnach dafür, dass ein definierter Prozentsatz von zufällig ausgewählten Neuronenaktivierungen einer Neuronenschicht während des Trainings automatisch auf 0 gesetzt wird. Dies hat laut Srivastava unter anderem den Effekt, dass der Koadaptation der Neuronenschichten untereinander entgegengewirkt wird. Koadaptation bedeutet hier, dass die Gewichte sich während des Trainings so anpassen können, dass die Aktivierung der Neuronen fehlerhafte Datentransformationen aus vorangestellten Schichten kompensieren können [53]. Da sich diese Koadaptationen nicht auf unbekannte Testdaten übertragen lassen, wurde dies als Ursache für Overfitting erkannt [53]. Hinzu kommen Methoden die sich speziell beim Einsatz in CNNs als hilfreich herausgestellt haben. Diesbezüglich betonen z. B. Ertel und Black der Gebrauch von Batch-Normalisierungen und Pooling-Schichten [41]. Neben einer üblichen Normalisierung der Eingangsdaten im Vorfeld des Trainings werden demnach die Daten mit der Batch-Normalisierung in sogenannten Batches (dt. Stapel) zwischen den Schichten zusätzlich standardisiert. In Pooling-Schichten wird eine Eingabe unter Verwendung eines Sliding-Window mit einem Pooling-Filter abgefahren [41]. Innerhalb des Fensterausschnitts wird anschließend ein Wert mithilfe einer deterministischen Funktion ermittelt. So wird z. B. der Maximal- („Max-Pooling“) oder Durchschnittswert („Average-Pooling“) innerhalb eines

Fensters erfasst (Abbildung 3-24) [41]. Die Schrittweite des Fensters entspricht dabei meist der Filterlänge, sodass es zu einer erheblichen Reduktion der Länge des berechneten Outputs kommt [41].

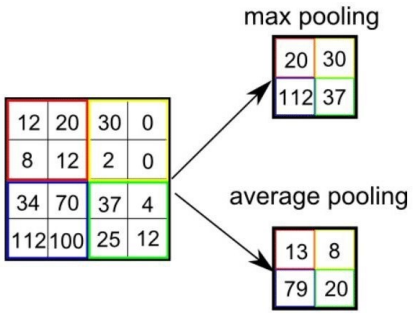


Abbildung 3-24: Gegenüberstellung „Max-Pooling“ und „Average-Pooling“ [54]

3.4 Netzarchitekturen tiefer neuronaler Netze

Im vorherigen Kapitel wurde die Funktionsweise von DNNs anhand der einfach verständlichen Architektur des MLP beschrieben. Für komplexe Problemstellungen werden jedoch meist deutlich anspruchsvollere Netzarchitekturen verwendet. Zudem kann sich die Repräsentation der Daten von Fall zu Fall unterscheiden, was wiederum die Anforderungen an die Struktur der neuronalen Netze verändert. Die Funktionsweisen weiterer für diese Arbeit betrachteten DL-Ansätze werden daher im Rahmen dieses Kapitels im Detail erklärt.

3.4.1 Eindimensionale Convolutional Neural Networks

Convolutional Neural Networks basieren im Allgemeinen auf einer Theorie, die bereits 1989 erarbeitet wurde [55]. Der Durchbruch von CNNs gelang schließlich durch den Fortschritt der verfügbaren Rechenkapazität im Bereich der Bilderkennung. Hauptsächlich hat dazu in 2012 der CNN-Ansatz AlexNet beigetragen, mit einer Rate von Falschklassifizierungen von 15,3 % von einem Bilddatensatz mit 1000 verschiedenen Klassen [56]. Dabei wurden die sehr umfangreichen Rechenoperationen beim Training des CNN hauptsächlich mithilfe von Graphical Power Units (GPU) umgesetzt. Neben dem Durchbruch für CNNs selbst, bedeutete dies ebenfalls einen Durchbruch für DNNs im Allgemeinen. Seitdem werden CNNs vor allem verstärkt im Bereich der Bildklassifizierung eingesetzt. Mit dem Aufschwung des DL durch AlexNet wurden CNNs in Form von eindimensionalen CNNs auch vermehrt zur Klassifizierung von Zeitreihendaten verwendet, wie z. B. im Bereich der Flugmanöverdetektion [15] [14]. Im folgenden Abschnitt wird daher die Theorie der CNNs am Beispiel von 1D-CNNs näher erläutert.

Eine Messreihe eines Sensors kann als eindimensionaler Vektor aufeinanderfolgender Messwerte (Zeitreihe) angesehen werden. Im realen Anwendungsfall werden solche Zeitreihen mit teils siebenstelligen Anzahlen von Messpunkten meist sehr lang und enthalten mehr als nur eine Klasse, die erkannt werden soll. Zur Klassifizierung von Zeitseriendaten, werden diese Datensätze häufig durch ein sogenanntes Sliding-Window einer bestimmten Größe und mit einer definierten Schrittweite sequenziert [57]. Dadurch wird immer nur ein Ausschnitt des Datenstroms in einem CNN verarbeitet. Die Messpunkte eines Sensors innerhalb eines solchen Sliding-Window werden als Inputvektor X an die Eingabeschicht des 1D-CNNs übergeben, wie Abbildung 3-25 entnommen werden kann.

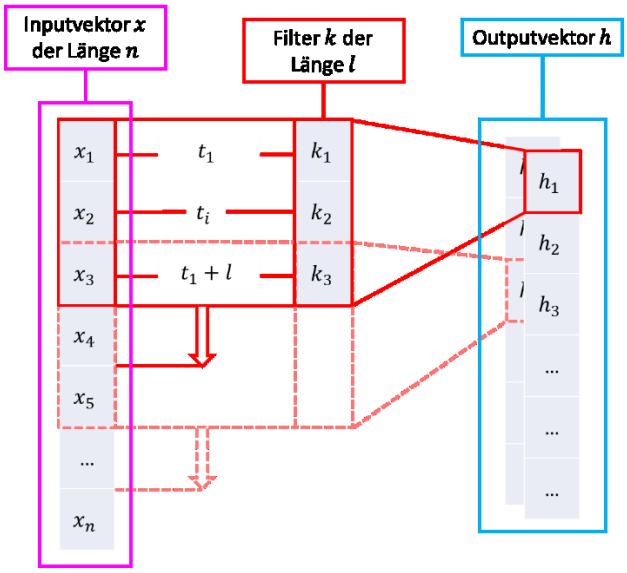


Abbildung 3-25: Schematische Abbildung einer 1D-Faltungsschicht (eigene Darstellung nach [58])

CNNs bestehen meist aus mehreren Arten funktionaler Schichten, die unterschiedliche Zwecke verfolgen. Dazu zählen sogenannte Faltungsschichten, Pooling-Schichten, Batch-Normalisierungs-Schichten und Dropout-Schichten. In einer Faltungsschicht wird erneut das Prinzip des Sliding-Window angewandt. Innerhalb eines Messdatenausschnitts fahren hierfür mehrere Filter k (auch: Kernel) mit fester Länge l die Messpunkte des Inputvektors X mit einer definierten Schrittweite s ab (Abbildung 3-25). Diese dienen dazu, verschiedene Merkmale im Abschnitt des jeweiligen Datenstroms zu erfassen. Ein Filter erfasst dabei immer jeweils ein Merkmal, das aus den Datenfenstern erfasst werden soll. [16]

Als bildliche Analogie zu einem MLP kann ein Filter als Gewichtsvektor angesehen werden, welcher zwei Neuronenschichten miteinander verbindet. Dementsprechend verläuft auch die Rechenoperation ab, sodass die Messwerte mithilfe der Berechnung des Skalarprodukts aus Inputvektor X und Filter k transformiert werden. Dieser Prozess der Merkmalserfassung wird Convolution (dt. Faltung) genannt, die der Netzarchitektur ihren Namen verleiht. Dadurch

entsteht in der nachfolgenden Schicht stets ein Outputvektor h pro verwendetem Filter k. Der Output an der Stelle t berechnet sich dazu wie folgt:

$$h_t = (x * k)_t = \sum_{\tau=t}^{t+l} \sum_{i=1}^{l} x_\tau * k_i \qquad 3.11$$

Da die Filter als Gewichtsvektoren eines klassischen Neuronalen Netzes zu verstehen sind, können dessen Werte ebenso durch Training angepasst werden. Das heißt während des Trainingsprozess wird durch die Anpassung der Filter automatisch gelernt, welche Merkmale aus den Daten für eine treffsichere Vorhersage des CNNs erfasst werden müssen. Beim Abtasten der Eingangsdatensequenzen bleiben die Gewichte der Filter zudem für jeden Zeitschritt gleich. Verglichen zur klassischen MLP-Architektur eines ANN kann dadurch die Anzahl der trainierbaren Parameter pro Neuronenschicht deutlich verringert werden, woraus wiederum ein geringerer Rechenaufwand resultiert. Aus Abbildung 3-25 wird zudem deutlich, dass die Eingangsdaten durch eine Faltungsschicht komprimiert werden. Falls der daraus resultierende Informationsverlust an den Rändern vermieden werden soll, ist es üblich den Eingangsvektor mit einem sogenannten „Zero Padding“ zu erweitern. Die Eingangsdatensequenz einer Faltungsschicht wird dabei an den Enden jeweils um einen oder mehrere zusätzliche Nullwerte erweitert. [58]

Hinzu kommt, dass Zustände im realen Anwendungsfall häufig nicht nur mit einem, sondern mit mehreren Sensoren gemessen werden, sodass die Messdaten in Form einer multivariaten Zeitserie vorliegen. In 1D-CNNs werden solche Messreihen durch mehrfache Input-Channels verarbeitet. Pro Merkmal wird dabei auf jeden Channel erneut jeweils ein Filter mit unterschiedlichen Filterwerten angewandt. Klassischerweise wird anschließend das Skalarprodukt aus allen Channels aufaddiert und an die Ausgabe übergeben. Allerdings existieren ebenfalls Ansätze, bei denen die Channel im Netz parallel und unabhängig voneinander weiterverarbeitet werden und zum Ende eines Blocks zusammengefügt werden. Bevor die transformierten Werte zur Klassifizierung an eine vollvernetzte Schicht übergeben werden, werden diese meist in einer Pooling-Schicht komprimiert. [58]

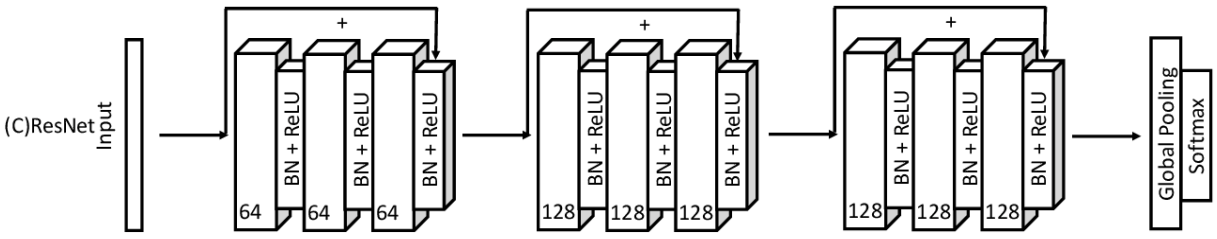


Abbildung 3-26: ResNet für die Zeitserienklassifizierung [57]

Eine spezielle Form der CNNs hat sich aus der Erkenntnis heraus entwickelt, dass CNNs für die Bildklassifizierung mit steigender Netztiefe immer bessere Ergebnisse erzielen können, bis die Leistungsfähigkeit ab einer gewissen Netztiefe fällt. Die Rede ist von den Residual Neural Networks, kurz ResNet [59]. Als Grund dafür, dass sich die Vorhersagen ab einer gewissen Netztiefe verschlechtern, wurde das Problem des „Vanishing/Exploding Gradient“ (dt.: verschwindender oder explodierender Gradient) identifiziert. Gemeint sind damit die Gradienten zur Anpassung der Gewichte und Bias, welche bei der Backpropagation mit erhöhter Anzahl der Schichten gegen null oder unendlich laufen. Somit erhöht sich das Risiko, dass sich Trainingszeiten stark erhöhen oder das Training stagniert bzw. das globale Minimum der Fehlerfunktion aufgrund zu hoher Gradienten nicht gefunden werden kann. Dies führt außerdem dazu, dass sich die Leistungsfähigkeit des Netzes bei langen Sequenzen stark verringert [60]. Um dies zu verhindern, verwendet ResNet sogenannte „Residual Connections“ (auch: „Shortcut Connections“ oder „Skip Connections“), um Ketten, welche die Gradienten durchlaufen bis zur Eingangsschicht zu verkürzen [59]. Dafür sind die Netze in sogenannten „Residual Blocks“ aufgebaut, die mehrere Faltungsschichten, Schichten zur Batch-Normalisierung und ReLU-Aktivierungsschichten zusammenfassen (Abbildung 3-26). Die „Residual Connections“ dienen dabei zur Addition der Eingangsdaten eines Blocks auf dessen Ausgangsdaten [59]. Das bietet den Gradienten die Möglichkeit die Blöcke bei der Backpropagation gewissermaßen zu umgehen und die Kette der partiellen Ableitungen. Wie bereits die CNNs wurde ResNet zu Beginn ebenfalls vordergründig für die Bildklassifizierung eingesetzt. Jedoch hat sich der Ansatz mittlerweile auch für die Zeitserienklassifizierung als verlässliche Lösung bewährt [16].

3.4.2 Long Short-Term Memory RNNs

Long Short-Term Memory Netze basieren auf Recurrent Neural Networks. Diese sind darauf spezialisiert sequenzielle Daten zu verarbeiten. Anders als bei klassischen ANNs ist dabei die Reihenfolge der Eingangsdatensequenz von Interesse. Denn RNNs zeichnen sich durch eine rekursive Struktur aus, wobei sie neben einem aktuellen Datenpunkt einer Sequenz ebenso die Ausgabe des vorherigen Datenpunkts als Input verwenden. Die Ausgabe aus einem vorherigen Neuron wird in diesem Fall auch als Hidden State h (auch: Short-Term Memory, dt.: verdeckter Zustand oder Kurzzeitgedächtnis) bezeichnet. h dient dabei einmal als Vorhersage für den jeweiligen Datenpunkt einer Sequenz und außerdem als Eingabe für das Neuron des nachfolgenden Datenpunktes. Abbildung 3-27 zeigt ergänzend dazu die grundlegende Funktionsweise von RNNs und wie sich diese bei Eingabe einer Datensequenz entfalten. Da zeitsequenzielle Daten mit dieser Architektur sehr gut verarbeitet werden können,

eignen sie sich beispielsweise als Lösungsansatz für ein Problem der Zeitserienklassifizierung. [61]

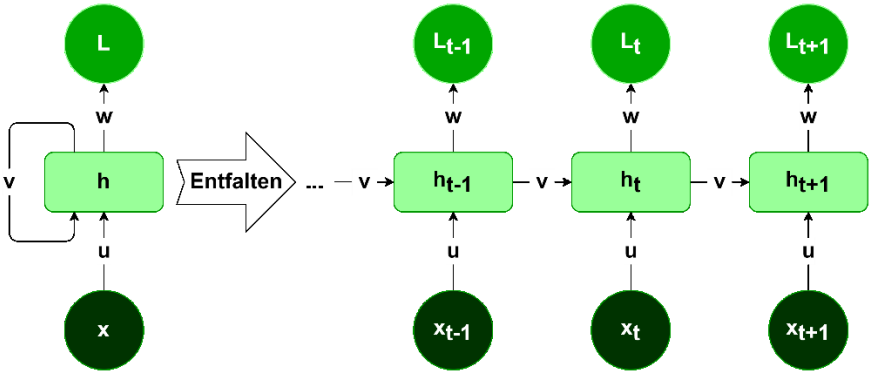


Abbildung 3-27: Funktionsweise von RNNs für Zeitserien nach [61]

Ein wesentliches Problem bei herkömmlichen RNNs ist allerdings ebenfalls der „Vanishing/Exploding Gradient“. Dieser tritt bei der Verarbeitung langer Zeitsequenzen dadurch auf, dass der Hidden State für jeden Zeitschritt mit einer Gewichtsmatrix v multipliziert wird [42]. Hier kommen LSTM-Netzwerke ins Spiel, die mit dem Langzeitgedächtnis als sogenannten Cell State c (dt.: Zellzustand, auch: Long-Term Memory) eine neue Größe einführen. Dieser wird gemeinsam mit dem Hidden State h von Zeitschritt zu Zeitschritt weitergegeben. Statt mithilfe einer Multiplikation, können c neue Informationen zum Langzeitgedächtnis durch Addition oder Subtraktion hinzugefügt bzw. abgezogen werden [42]. Dies vermindert den Effekt, dass die Gradienten mit erhöhter Sequenzlänge, aufgrund der häufigen Transformation der Daten durch aufeinanderfolgende Multiplikationen, gegen null oder unendlich laufen. LSTMs wurden erstmals bereits 1997 von Hochreiter und Schmidhuber eingeführt [62]. Sie sorgten laut Russel und Norvig dafür, dass RNNs für komplexe Problemstellungen überhaupt erst anwendbar wurden [42]. So konnten sie z. B. erfolgreich bei Aufgaben wie der Spracherkennung und der Handschrifterkennung eingesetzt werden. Gegenüber klassischen RNNs erfassen LSTMs Zusammenhänge in höherer Komplexität und vor allem über einen längeren Zeitraum hinweg, wodurch sie mittlerweile als populärer Lösungsansatz für die Analyse von Zeitreihendaten gelten. So auch im Bereich der Klassifizierung und Vorhersage von Flugtrajektorien und Flugmanövern [18] [63].

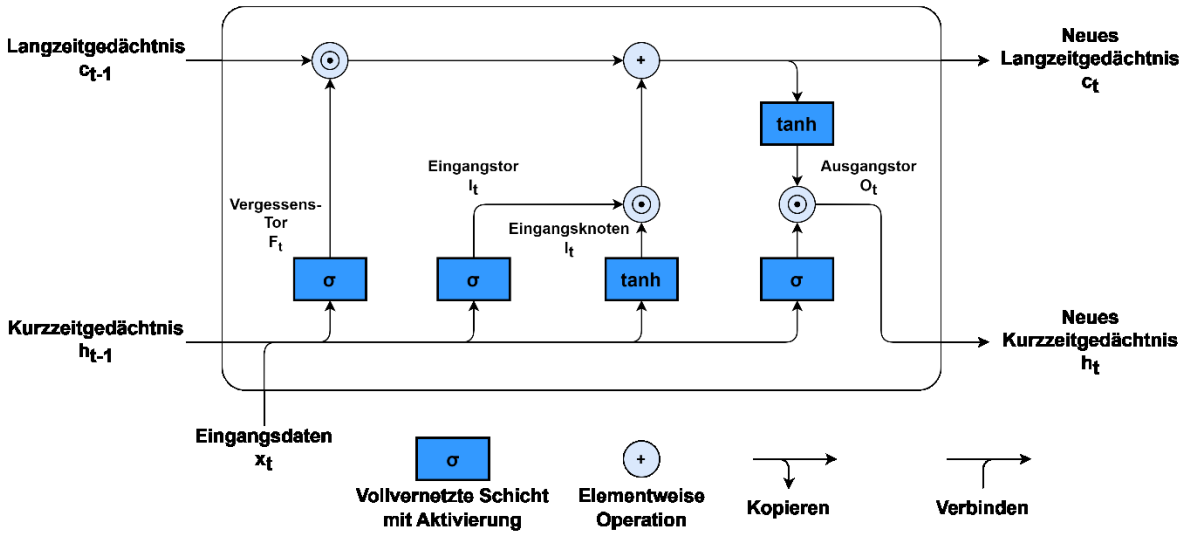


Abbildung 3-28: Funktionsweise und Datenfluss in einer LSTM-Zelle nach [64]

LSTM-Netze nutzen im Gegensatz zum klassischen Ansatz komplexere Einheiten anstelle der bisher beschriebenen Neuronen, in denen zahlreiche Transformationen der Daten stattfinden. Zur Verdeutlichung des Datenflusses und der Funktionsweise von LSTM-Zellen dient an dieser Stelle Abbildung 3-28. Analog zum klassischen Ansatz nutzen LSTM-Netze den Hidden State eines vorherigen Datenpunktes h_{t-1} um Vorhersagen für einen Datenpunkt x_t zu treffen. Verglichen zu RNNs kommen in diesem Fall spezielle Mechaniken zur Speicherung von Langzeitinformationen hinzu. Diese werden durch drei sogenannte Tore realisiert: das Eingangstor (engl.: input gate), das Ausgangstor (engl.: output gate) und das Vergessens-Tor (engl.: forget gate). Dazu nutzen die Zellen die Sigmoid- und tanh-Aktivierungsfunktion, um zum einen bestimmte Prozentsätze vorhandener Informationen im Cell State c zu speichern und zum anderen, um c Informationen hinzuzufügen oder abzuziehen. Jede LSTM-Zelle berechnet also einen neuen Zustand c_t der Zelle und eine neue Ausgabe h_t basierend auf dem vorherigen Zustand c_{t-1}, der vorherigen Ausgabe h_{t-1} sowie dem aktuellen Input x_t. Das Eingangstor entscheidet dabei, welcher Anteil neuer Informationen von x_t und h_{t-1} zum Langzeitgedächtnis hinzugefügt oder davon abgezogen werden soll. Zuvor bestimmt das Vergessens-Tor, welcher Anteil vom bisherigen Langzeitgedächtnis c_{t-1} weiterverwendet werden soll. Die Aktivierungsfunktionen der verschiedenen Tore nutzen dabei die Zusammensetzung aus dem jeweiligen Datenpunkt der Eingangsdatensequenz x_t und der Ausgabe des Ausgangstors in der vorherigen LSTM-Einheit h_{t-1} als Eingangsvariable. Der neue Wert des Langzeitgedächtnisses c_t definiert abschließend gemeinsam mit einem Anteil der neuen Informationen aus h_{t-1} und x_t durch das Ausgangstor den Wert des neuen Hidden State h_t. Diese Struktur ermöglicht es LSTM-Netzwerken Informationen über längere

Zeiträume hinweg zu speichern und somit relevante Muster und Merkmale aus früheren Sequenzen für Vorhersagen oder Klassifizierungen wiederzuverwenden. [65]

3.5 TWINSTASH

Die Grundlage für den zuvor eingeführten *twinstash* bilden die DLR-Projekte DigTwin (Digital Twin, 2018-2022) [66] und DigECAT (2022-2025) [8]. In DigTwin wurden zunächst eine Roadmap sowie die Rahmenbedingungen und Anforderungen zur Umsetzung digitaler Zwillinge von der DLR-Forschungsflotte erarbeitet. Das bis 2025 laufende DigECAT-Projekt befasst sich darauf aufbauend mit der konkreten Umsetzung eines digitalen Zwillings zunächst auf Komponenten- und Systemebene und später plangemäß ebenfalls auf Flugzeugebene. Im Zuge dessen sollen nicht nur der digitale Zwilling selbst, sondern darüber hinaus Methoden, Prozeduren und Werkzeuge zur Arbeit mit digitalen Zwillingen entwickelt und standardisiert werden. In diesem Zusammenhang stellt der *twinstash* einen wichtigen Baustein dar. Als Service Plattform zur Bereitstellung der Luftfahrzeuginformationen soll diese verschiedenen Instituten und Einrichtungen unterschiedlicher Disziplinen zum einen als zentrale Daten- und Informationsquelle dienen [66] [9]. Zum anderen sollen im *twinstash* Anwendungen und Werkzeuge zur Verfügung gestellt werden, welche selbigen Instituten und Einrichtungen als Hilfsmittel zum weiteren Informationsgewinn aus den Daten dienen sollen. Dafür stehen den Benutzern sowohl ein Graphical User Interface (GUI, dt.: graphische Benutzerschnittstelle) in Form einer Web-Oberfläche (Abbildung 3-29) als auch ein Application Programming Interface (API, dt.: Anwendungsprogrammierschnittstelle) in Form eines Python-Clients zur Verfügung. Zum aktuellen Zeitpunkt steht die Plattform lediglich ausgewählten Instituten und Einrichtungen des DLR zur Verfügung und wird in Zusammenarbeit mit diesen kontinuierlich weiterentwickelt. In der aktuellen Version stellt der *twinstash* bereits eine große Menge an gesammelten Flugmessdaten verschiedener Forschungsflugzeuge des DLR zur Verfügung [9]. Zur Interaktion mit den Messdaten stehen den Benutzern dafür in der Web-Oberfläche bereits Visualisierungs- und Analysetools zur Verfügung, die beispielsweise die Darstellung der 2D- und 3D-Trajektorien des Flugzeugs ermöglichen [9]. Des Weiteren wird den Benutzern der Zugriff auf die Messsignale verschiedener Flüge ermöglicht. Ziel ist es in zukünftigen Versionen u. a. Funktionen zur Verfügung zu stellen, die zu einer aussagekräftigen Beschreibung der Ontologie und Semantik von Daten auf dem *twinstash* beitragen [8]. Als Grundlage dafür stehen bereits Modelle zur Überwachung des Sensorzustands, zur Strukturüberwachung sowie zur Flugphasenerkennung zur Verfügung.

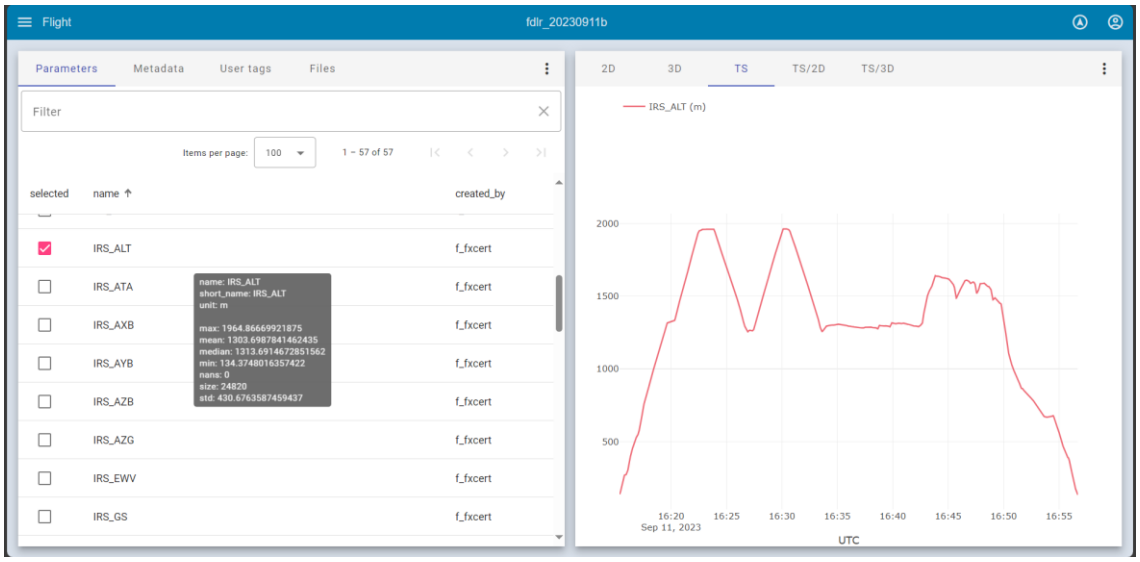


Abbildung 3-29: Datenverlauf Flughöhe *twinstash* Web-Oberfläche

4 Vorgehensweise nach MLOps

Zur kontinuierlichen Integration und Entwicklung von Machine Learning Ansätzen wurde das Konzept der Machine Learning Operations (MLOps, dt.: Abläufe des Maschinellen Lernens) von Kreuzberger et al. erarbeitet [67]. Damit wird eine Richtlinie zur Verfügung gestellt, die „Aspekte wie Best Practices, Konzepte sowie eine Entwicklungskultur bezüglich der End-to-End-Konzeptionierung, Implementierung, Überwachung, dem Einsatz und der Skalierung von ML Produkten umfasst“ [67]. Dies ermöglicht demnach eine Fehlerreduzierung dieser Algorithmen und damit ebenfalls einen schnelleren Einsatz in den Wertschöpfungsketten. Diese Prinzipien von MLOps werden in einer End-to-End Architektur bzw. einem Workflow verankert (Abbildung 4-1). Der Workflow reicht von der Konzepterstellung eines ML-Produkts (A) über die Pipelines zur Datenvorbereitung (B1 + B2) und zum Modelltraining (C) bis hin zum Einsatz des ML-Modells in den Wertschöpfungsketten. Letzteres umfasst ebenfalls eine dritte Pipeline zur automatisierten Optimierung der ML-Modelle im Einsatz, welche für die vorliegende Arbeit allerdings keine Rolle spielt. Stattdessen beschränkt sich diese Arbeit auf die Bereiche der Konzeptionierung von DL-Ansätzen, der Datenvorbereitung und dem Prozesskette des Modelltrainings in MLOps.

(A) Konzeptionierung eines Deep-Learning-Ansatzes

Die Konzeptionierung des Deep Learning Modells startet mit der Analyse des Anwendungsfalls, in dem ein Problem erkannt wird, das mithilfe eines DL-Algorithmus gelöst werden kann. Daraufhin werden die Modellarchitektur und bekannte oder neue Ansätze definiert, die zur Lösung des vorliegenden Problems verwendet werden können. Dafür wird der Stand der Forschung als Bewertungsgrundlage hinzugezogen. Im nächsten Schritt werden die betrachteten DL-Ansätze durch die Definition der Methodik eingegrenzt. So wird zum Beispiel entschieden, ob ein vorliegendes Problem mit einem Klassifizierungsansatz oder einer Regressionsanalyse gelöst werden soll. Anschließend wird eine Anforderungsliste für die Daten erstellt, welche zum Training eines entsprechenden Modells benötigt werden. Dies beinhaltet nicht nur die Frage nach der Qualität der Daten, sondern zudem, ob diese beispielsweise schon vorhanden sind bzw. woher diese zu erhalten sind. Dementsprechend müssen eventuelle Maßnahmen ergriffen und Kampagnen koordiniert werden, um z. B. neue Daten zu erzeugen. Sind die Daten vorhanden, können diese analysiert werden. Dies umfasst z. B. die Beschriftung der Daten im Rahmen eines Daten Labeling Prozesses. Werden geforderte Datenkriterien nicht erfüllt, kann es vorkommen, dass der Prozess der Datenakquise und -analyse in mehreren Iterationsschleifen bis zu einer hinreichenden Datenqualität durchlaufen werden muss. [67]

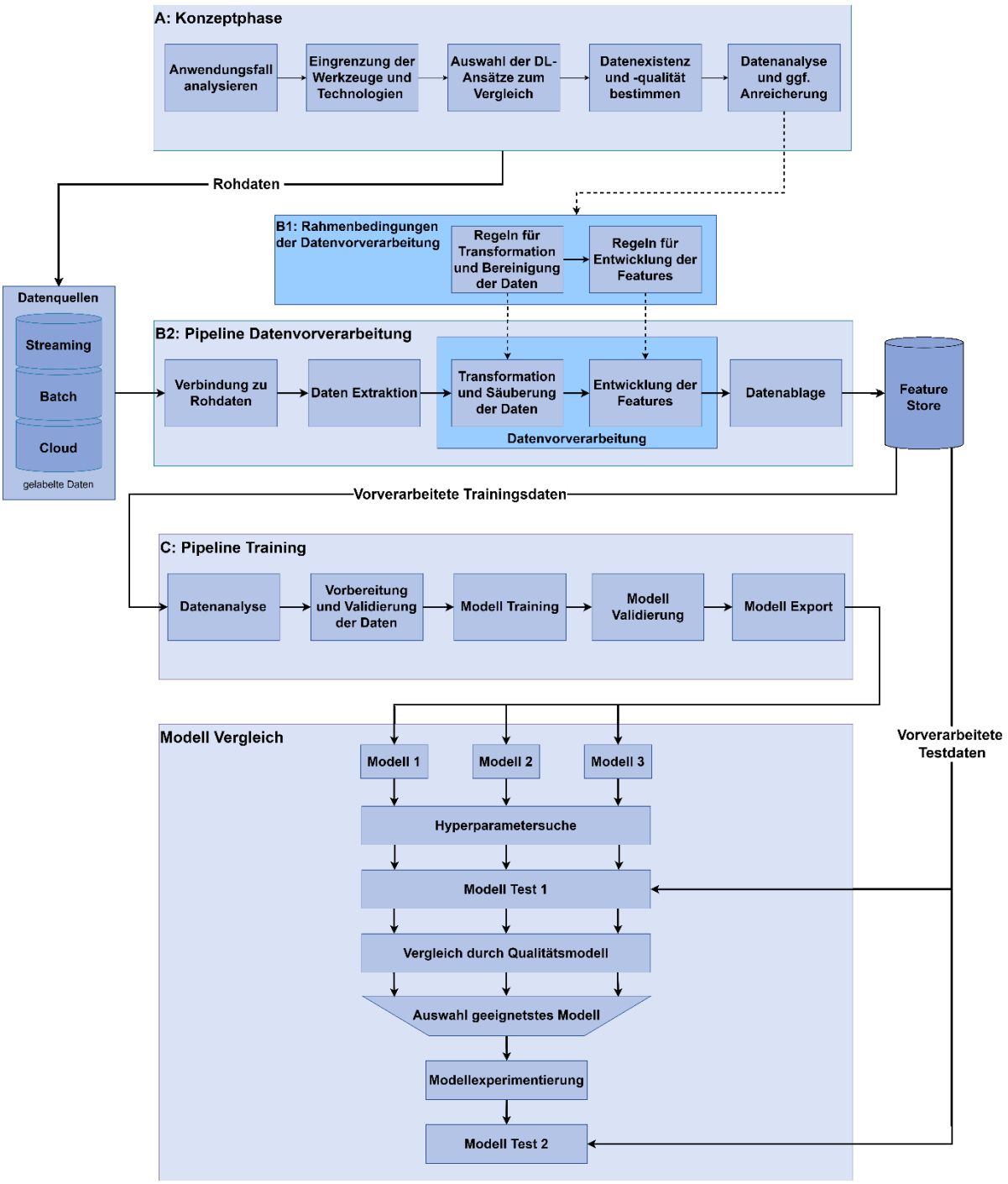


Abbildung 4-1: Workflow zur Entwicklung von DL-Ansätzen nach MLOps [67]

(B) Pipeline zur Datenvorverarbeitung

Bevor der daraus resultierende Datensatz in die Pipeline zur Datenvorbereitung eingespeist wird, wird der Kenntnisstand über den Datensatz zunächst benutzt, um daraus notwendige Rahmenbedingungen für die Datentransformation und die Entwicklung von Merkmalen abzuleiten. Dabei wird beispielsweise entschieden, ob die Daten normalisiert oder standardisiert werden, falls es sich um einen Datensatz mit vielen oder wenigen enthaltenen

Ausreißern handelt. Sind die Rahmenbedingungen festgelegt, kann die Pipeline zur Datenvorbereitung an die Datenquelle angeschlossen werden. Anschließend werden die Daten extrahiert und falls nötig in das korrekte Format konvertiert. Ist dies erfolgt können die Daten anhand der erarbeiteten Rahmenbedingungen transformiert und bereinigt werden. Dies beinhaltet z. B. Schritte zur Bereinigung von Duplikaten oder zur Standardisierung der Daten. Daraufhin wird laut MLOps regulär der Schritt des Feature Engineering angewandt, um komplexere Merkmale in den Daten zu spezifizieren. Abschließend wird der neu angepasste Datensatz in einem sogenannten Feature Store System abgelegt und an die Trainingspipeline übergeben. Für das Training des zu entwickelnden Modells (C) können an dieser Stelle die transformierten, bereinigten und gelabelten Datensätze bezogen werden. [67]

(C) Pipeline zum Training und zur Optimierung eines DL-Modells

Zu Beginn des Modelltrainings werden die vorverarbeiteten Datensätze aus dem Feature Store System extrahiert und - falls notwendig - erneut analysiert, um potenzielle Anpassungsmaßnahmen an die Datenvorverarbeitung zurückzumelden. Daraufhin werden im Schritt der Datenvalidierung und -vorbereitung die vorverarbeiteten Trainingsdaten unter anderem in Trainings- und Validierungsdatensatz aufgeteilt. Im nächsten Schritt werden die optimalen Algorithmen sowie die passenden Hyperparameter abgeschätzt und eingestellt. Damit wird der Trainingsprozess eingeleitet, in dem anfangs das Modell trainiert wird, bevor die Trainingsergebnisse anhand der Validierungsergebnisse evaluiert werden und daraus Anpassungen der Modellparameter abgeleitet werden. Mit den neuen Modellparametern wird daraufhin eine neue Iteration durchlaufen. Erkenntnisse aus diesem Zyklus werden zudem verwendet, um Verbesserungsmaßnahmen an den Rahmenbedingungen für die Datenvorverarbeitung vorzunehmen und die Daten gegebenenfalls erneut anzupassen. Die Iterationsschleife wird solange wiederholt, bis die Leistungsmetriken die gewünschten Ergebnisscores aufweisen. [67]

Da die vorliegende Arbeit verschiedene DL-Ansätze miteinander vergleicht, werden die Modelle der unterschiedlichen Ansätze anschließend im Rahmen eine Parametersuche optimiert, getestet und miteinander verglichen. In einer Experimentierphase sollen anschließend Verbesserungsansätze für den DL-Ansatz identifiziert werden, der aus dem Vergleich als am geeignetsten hervorgeht.

5 Vergleich Deep-Learning-Ansätze

Auf Grundlage des in Kapitel 4 beschriebenen Vorgehensmodells wird in diesem Abschnitt die Entwicklung der DL-Algorithmen erläutert. Dazu wird zunächst das Konzept der DL-Ansätze zur automatisierten Erkennung von Flugmanövern auf Basis von Flugdaten erarbeitet. Dies umfasst unter anderem die Analyse des Anwendungsfalls Flugmanöverdetektion sowie die Vorgehensweise beim Labeling der verfügbaren Flugdaten. Zudem werden das Konzept sowie die Auswahl der Qualitätsmaßstäbe zur Bewertung der DL-Modelle beschrieben. Anschließend werden die Schritte zur Vorverarbeitung der Datensätze erläutert. Im nächsten Schritt wird die Modelltrainingspipeline für drei verschiedene DL-Modelle konfiguriert. Nach dem Training der Modelle wird eine Parametersuche für alle Ansätze durchgeführt. Für einen qualitativen Vergleich werden die Ergebnisse anschließend in einer Nutzwertanalyse verwendet. Darauf aufbauend wird ein Ansatz für eine weitere Optimierung ausgewählt, weiterentwickelt und abschließend erneut evaluiert.

5.1 Konzeptphase automatisierte Manöverdetektion

Anwendungsmöglichkeiten und Eigenschaften von digitalen Zwillingen im Allgemeinen wurden in Kap. 1 bereits kurz vorgestellt. Als ein wichtiger Aspekt wurde dabei der hohe Informationsumfang angeführt, den ein digitaler Zwilling über ein physisches Objekt im digitalen Raum bereitstellen soll. Speziell in Verbindung mit dem Flugversuch sind die Datenabschnitte, in denen Testmanöver geflogen wurden, von besonderem Interesse für die Flugversuchsingenieure. Eine Kennzeichnung der Datenabschnitte mit den entsprechenden Manövern machen die Flugdaten im Allgemeinen durchsuchbar und die Manöver somit leichter auffindbar. Aufgrund der Notwendigkeit der Datenkennzeichnung, handelt sich also allgemein um ein Problem der Klassifizierung von Zeitreihendaten. Je nach Anwendungsfall unterscheidet sich die Metrik, die dazu vordergründig zur Bewertung der Leistungsfähigkeit eines Klassifikators verwendet werden sollte. Die Definition der Metrik ist daher eng an die Intention der Manövererkennung gekoppelt.

5.1.1 Anwendungsfall Manöverdetektion

Kap. 3.2.2 hat bereits dargelegt, wie Flugmanöver zur Ermittlung der Flugeigenschaften im Rahmen des Flugversuchs eingesetzt werden. In diesem Zusammenhang bieten sich verschiedene Anwendungsmöglichkeiten einer automatisierten Flugmanöverdetektion. Die Reaktionen eines Flugzeugs können sich bei variierenden Iterationen eines Manövers stark unterscheiden. Dementsprechend sind manche Flugmanöver kritischer für die Ausprägung der

Flugeigenschaften als andere. Ein möglicher Anwendungsfall wäre daher, die Durchsuchbarkeit der Messdaten infolge einer Anreicherung der Metadaten zu nutzen, um anhand der Datenlage eines Manövers Informationen über potenzielle Flugeigenschaften eines Flugzeugs bereitzustellen. Des Weiteren können die angereicherten Flugdaten infolge einer Flugmanöverdetektion dazu genutzt werden, um Flugmanöver in autonomen Flugsteuerungssystemen zu trainieren. Dies ermöglicht es den Systemen sich an bestimmte Flugbedingungen und Flugszenarien anzupassen und so einen sicheren Flugbetrieb zu gewährleisten. Hinzu kommt die Möglichkeit historische Analysen von bestimmten Flugmanövern anzufertigen, woraus Trends und Muster erkannt werden können. Diese tragen wiederum dazu bei, Simulationen realitätsgetreuer zu gestalten. Für all diese Anwendungsfälle ist es wichtig sicherzustellen, dass alle relevanten Daten von Flugmanövern in den Daten vorhanden sind. Speziell im Fall der automatisierten Ermittlung von Flugeigenschaften und zur Entwicklung autonomer Flugsteuerungssysteme ist dieses Kriterium von vorrangiger Bedeutung, da dies sicherheitskritisch sein kann. Dabei soll sichergestellt werden, dass nahezu alle wirklich geflogenen Manöver erkannt werden, sodass kritische Manöver nicht übersehen werden und die Datenbasis so umfassend und repräsentativ wie möglich ist.

5.1.2 Auswahl Bewertungskriterien

Um zu prüfen, ob die Modelle die oben beschriebenen Anforderungen erfüllen, muss deren Qualität bewertet werden. Die vorliegende Arbeit orientiert sich dafür an dem Qualitätsmodell von Siebert et al., da es neben der reinen Leistungsfähigkeit eines Modells noch weitere Aspekte wie z. B. die Stabilität eines Modells berücksichtigt [23]. Die Auswahl der Bewertungskriterien orientiert sich demnach an verschiedenen Normen und dem Fachwissen der Autoren. Tabelle 5-1 stellt eine Auflistung und Beschreibung der verschiedenen Bewertungskriterien des Qualitätsmodells bereit, die in dieser Arbeit verwendet werden.

Aufgrund der in Kap. 5.1.2 beschriebenen Anforderungen an die DL-Modelle, wird für die Bewertung der Leistungsfähigkeit der Modelle vorrangig die Sensitivität bzw. der Recall (Kap. 3.3.1) verwendet. Grund dafür ist, dass der Recall umso höher ausfällt, je geringer der Anteil falsch negativer Vorhersagen eines Klassifikators ausfällt (Gl. 3.8). Die sog. Precision ist in diesem Fall unkritischer und daher bei der Bewertung von nachrangiger Bedeutung. Eine niedrige Precision bei hohem Recall würde dementsprechend Folgendes bedeuten: zwar wurde in dem Fall nur ein geringerer Anteil der Vorhersagen eines Manövers auch wirklich geflogen, dafür decken die positiven Vorhersagen jedoch alle wahren Flüge des Manövers mit ab. Bei der Ermittlung der Flugeigenschaften wird so beispielsweise die Envelope möglicherweise stärker eingegrenzt als nötig, allerdings wird ein Sicherheitsrisiko somit

definitiv vermieden. Da es sich außerdem, um eine Mehrklassenklassifizierung mit unausgewogenen Daten handelt werden zusätzlich der Recall pro Klasse, die Mehrklassen-Wahrheitsmatrix und der Matthews Correlation Coefficient in die Auswertung der Ergebnisse miteinbezogen.

Tabelle 5-1: Bewertungskriterien für die Modellqualität eines ML-Algorithmus [23]

Kriterium	Beschreibung	Bewertungsmaßstab
Korrektheit im Entwicklungsprozess	Leistungsfähigkeit des Modells auf den Trainings- und Validierungsdaten die geforderte Aufgabe zu erfüllen	Recall Validierung [23]
Korrektheit im Einsatz	Leistungsfähigkeit des Modells auf den Testdaten die geforderte Aufgabe zu erfüllen	Recall Test [23]
Relevanz	Über- oder Unterangepasstheit eines Modells	Differenz Recall Validierung-Test
Robustheit	Anpassungsfähigkeit eines Modells, auch bei hoher Datenvarianz oder unvollständigen Datensätzen die geforderte Aufgabe zu erfüllen	Manöver Dutch Roll: klassenspezifischer Recall Validierung
Stabilität	Fähigkeit des Modells reproduzierbare Trainingsergebnisse mit verschiedenen Durchmischungen des Datensatzes zu erzielen	Standardabweichung Recall Validierung
Fairness	Voreingenommenheit des Modells, Unterscheidung zwischen einzelnen Klassen	Differenz zwischen max. und min. klassenspezifischen Recall
Interpretierbarkeit	Möglichkeit die Ergebnisse eines Modells zu begründen und die Entscheidungsfindung des Modells nachverfolgen zu können	Anzahl der Schichten, Anzahl der Neuronen/Filter pro Schicht, Anzahl der trainierbaren Gewichte [23]
Ressourcenaufwand	Ressourcen, die das Modell während des Trainings oder der Evaluation verbraucht.	Speicheraufwand, Trainingsdauer [23]

Die Relevanz eines Modells wird anhand der Differenz zwischen dem Recall aus Validierung und Test des jeweiligen Modells bewertet, sodass der Effekt einer potenziellen Überanpassung quantitativ in die Bewertung einfließen kann. Die Robustheit wird anhand des klassenspezifischen Recall-Scores für das Dutch Roll Manöver bewertet. Das Manöver wurde ausgewählt, da aus dem Labeling-Prozess (Kap. 5.1.3) bekannt war, dass die Signalverläufe des Dutch Roll Manövers durch den Einsatz des Gierdämpfers und durch die Störungsanfälligkeit eine hohe Varianz aufweisen. Die Stabilität des Modells wird anhand der Standardabweichung des Recall-Scores aus verschiedenen Validierungsdurchläufen bewertet. Die Bewertung der übrigen Kriterien orientiert sich an Siebert et al. [23].

Mithilfe der Methode des paarweisen Vergleichs werden die Kriterien gewichtet, sodass der Einfluss von irrelevanteren Kriterien auf die Bewertung geringer ist. Dafür werden sie in einer Kreuztabelle einzeln einander gegenübergestellt und als wichtiger (2), gleichwichtig (1) oder unwichtiger (0) bewertet. Die Punkte werden für jedes Kriterium einzeln aufsummiert und ergeben schließlich einen Score. Im Anschluss wird absteigend nach Score ein Ranking über die Relevanz der Kriterien für die vorliegende Arbeit erstellt und daraus eine Gewichtung der Kriterien abgeleitet. [68]

Wichtiger \ Als	Korrektheit im Entwicklungsprozess	Korrektheit im Einsatz	Relevanz	Robustheit	Stabilität	Fairness	Interpretierbarkeit	Ressourcenaufwand	Summe	%
Korrektheit im Entwicklungsprozess		0	0	0	0	2	2	2	**6**	**10.71%**
Korrektheit im Einsatz	2		1	1	1	2	2	2	**11**	**19.64%**
Relevanz	2	1		1	0	2	2	2	**10**	**17.86%**
Robustheit	2	1	1		1	2	2	2	**11**	**19.64%**
Stabilität	2	1	2	1		2	2	2	**12**	**21.43%**
Fairness	0	0	0	0	0		2	2	**4**	**7.14%**
Interpretierbarkeit	0	0	0	0	0	0		2	**2**	**3.57%**
Ressourcenaufwand	0	0	0	0	0	0	0		**0**	**0.00%**
								Prüfsumme		**100.00%**

Abbildung 5-1: Paarweiser Vergleich der Bewertungskriterien nach Siebert et al.

Daraus resultiert folgendes Ranking der Bewertungskriterien:

1. Stabilität 21,43 %
2. Korrektheit im Einsatz 19,64 %
3. Robustheit 19,64 %
4. Relevanz 17,86 %
5. Korrektheit im Entwicklungsprozess 10,71 %
6. Fairness 7,14 %
7. Interpretierbarkeit 3,57 %
8. Ressourcenaufwand 0,00%

Abbildung 5-1 zeigt den paarweisen Vergleich der ausgewählten Bewertungskriterien und die daraus hervorgehende Gewichtung. Als wichtigstes Kriterium geht die **Stabilität** aus dem paarweisen Vergleich hervor, da das gewählte Modell möglichst konstante Ergebnisse erzielen soll. Dicht darauf folgen die **Korrektheit im Einsatz und die Robustheit**, da das Modell eine gute Leistungs- und Generalisierungsfähigkeit aufweisen soll. Da es sich zunächst nur um ein Offline-Modell handelt, das die Infrastruktur der Einrichtung Flugexperimente zur Berechnung nutzen kann, wird der **Ressourcenaufwand** am schwächsten gewichtet und wird daher nicht weiter berücksichtigt. Die **Korrektheit im Entwicklungsprozess, Fairness** und die **Interpretierbarkeit** werden als mittelmäßig bedeutsam gewichtet.

5.1.3 Analyse Datensatz

Für diese Arbeit wurden Flugmessdaten aus Flugversuchen verwendet, die in den Jahren 2016, 2018, 2019, 2020, 2022 und 2023 im Rahmen der jährlichen „DLR Summer School“ durchgeführt wurden. Dabei durchlaufen ausgewählte Studenten unterschiedlicher Universitäten und Hochschulen einen typischen Systemidentifizierungsprozess für ein Flächenflugzeug. Dies umfasst unter anderem die Durchführung von Flugversuchen zur beispielhaften Untersuchung der Flugzeugeigenschaften und die Bestimmung bestimmter Flugeigenschaften des Flugzeugs auf Grundlage der Versuchsmessdaten. Im Rahmen dieses Projektes werden viele typische Flugmanöver durchgeführt, die sich in weiteren Messkampagnen mit anderen Forschungsflugzeugen des DLR wiederfinden. Daher eignen sich diese Daten gut, um die Anwendbarkeit eines DL-Modells auf typische Flugmanöver in der Flugerprobung zu bewerten. Die im Rahmen dieser Arbeit verwendeten Flugdaten erstrecken sich vom September 2016 bis zum September 2023. In diesem Zeitraum werden Rohdaten aus insgesamt 26 Versuchsflügen betrachtet, die kumuliert 28:47 Stunden an

Messdaten liefern. Die Rohdaten liegen im „Network Common Data Form“-Format (netCDF) vor, welches ein systemunabhängiges Dateiformat ist und die Erstellung, den Zugriff und das Teilen von wissenschaftlichen Messreihendaten unterstützt [69]. Die Extraktion der Messwerte und deren Konvertierung zu NumPy-Arrays wurde im Rahmen der vorliegenden Arbeit mithilfe der Python-Bibliothek „netCDF4“ in einem Parser in Form eines Python-Scripts realisiert. Insgesamt nehmen die netCDF-Dateien eine Speicherplatzmenge von 1,36 Gigabyte (GB) ein. Bei der Verarbeitung der Flugdaten traten keine Probleme auf, sodass diese vollständig verarbeitet werden konnten. Für alle Flüge zusammen konnte eine multivariate Zeitserie mit 58 verschiedenen Messreihen erfasst werden. Aufgrund der Weiterentwicklung der Messanlage über die Jahre mit zusätzlichen Sensoren, liegen bei späteren Versuchsflügen mehr Messreihen vor. Daraus resultiert, dass 13 Messreihen zwischen 52.140 und 6.225.170 leere Messwerte enthalten. Der Umgang mit diesen Datenpunkten wird im späteren Verlauf dieses Kapitels erläutert. Hinzu kommt die Flugbezeichnung, sodass aus den Versuchsflügen zunächst ein Datensatz mit insgesamt 59 Merkmalen hervorgeht. Zwar wurden je Flug alle Messreihen mit derselben Messfrequenz aufgenommen, jedoch unterscheiden sich diese zum Teil von Flug zu Flug. Insgesamt wurden neun Flüge mit einer Sampling Rate von 10 Hz, ein Flug mit einer Sampling Rate von 30 Hz und die restlichen 16 Flüge mit einer Sampling Rate von 100 Hz aufgenommen. Daraus resultiert eine Anzahl von 6.582.770 Messpunkten, die insgesamt in allen Versuchsflügen aufgezeichnet wurden. Diese Flugdaten sollen als Datengrundlage für das Training und Testen der DL-Ansätze dienen. Das Testen der Netze wird mit zwei im Datensatz enthaltenen Flügen separat durchgeführt, die zur Vorbereitung der „DLR Summer School“ im Jahr 2020 aufgenommen wurden. Dabei handelt es sich um zwei Musterflüge, in denen alle Manöver, die im Rahmen dieser Arbeit untersucht werden, mindestens einmal vertreten sind. Beide Musterflüge wurden mit einer Datenfrequenz von 100 Hz aufgenommen und umfassen insgesamt 3:12 Stunden an Flugmessdaten. Daraus ergeben sich insgesamt 1.156.100 aufgenommene Messpunkte. Damit machen die Testdaten einen Anteil von 18 % des gesamten Datensatzes aus (Abbildung 5-2). Pro Messpunkt sind ebenfalls 58 verschiedene Messsignale aufgenommen worden. Davon enthalten in diesem Fall allerdings vier Signale keine Werte, weshalb nur 54 Messignale effektiv nutzbar sind. Zusammen mit dem Parameter der Flugbezeichnungen liegt somit ebenfalls ein Datensatz mit 59 Parametern pro Messpunkt vor.

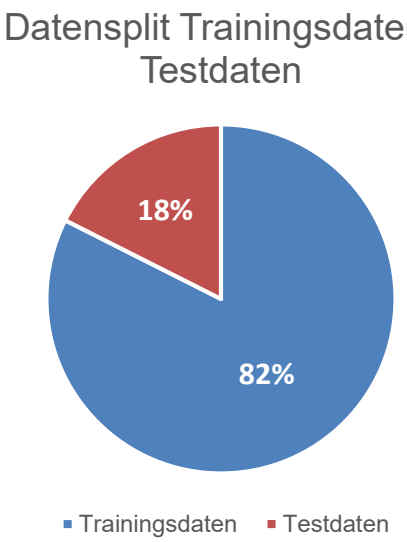


Abbildung 5-2: Datensplit Trainingsdaten-Testdaten

Als Grundlage für das überwachte Lernen neuronaler Netze dienen gelabelte Daten. Da die Manöver in dem vorhandenen Flugmessdatensatz noch nicht gelabelt sind, wird dieser Schritt manuell durchgeführt. Insgesamt werden in den Flugmessdaten vier komplexe Manöver gelabelt. Das sind die Manöver Dutch Roll, Steady Pull-Up, Steady Push-Over und Elevator Sweep. Diese Manöver wurden ausgewählt, da sie zum einen in nahezu allen Flügen des „DLR Summer School“ Projekts geflogen wurden und somit in ausreichend hoher Anzahl vertreten sind. Zum anderen handelt es sich hierbei um gängige Manöver der Flugerprobung, die ebenfalls im Rahmen anderer Messkampagnen geflogen werden und daher eine projektübergreifende Relevanz besitzen. Da die Flugmanöver jedoch nur einen geringen Anteil der gesamten Flugmessdaten ausmachen, würde durch das alleinige Labeln der komplexen Manöver ein starkes Klassenungleichgewicht gegenüber den übrigen Messdaten auftreten. Dies kann nachteilig für die Leistungsfähigkeit der Modelle sein. Daher werden zusätzlich sechs einfache Flugzustände in den Messdaten markiert, die mit Sicherheit in nahezu jedem Flug in ausreichender Häufigkeit durchgeführt werden. Dazu gehören der Abflug (Take-Off), der Steigflug (Climb), der Sinkflug (Descent), die Kurvenflug links (Turn Left), die Kurvenflug rechts (Turn Right) und der Landeanflug sowie die Landung (Descent, Approach, Landing). Damit werden insgesamt zehn Manöver in den Daten gelabelt. Alle Flugabschnitte, in denen keine Manöver durchgeführt oder erkannt werden, werden als „Unclassified“ (dt.: nicht klassifiziert) gekennzeichnet. Die Kriterien zum Labeling der Manöver werden von den Manövereigenschaften abgeleitet, die bereits in Kap. 3.2.2 erläutert wurden. Während des Labelingprozesses fällt auf, dass die Manöver Take-Off, Dutch Roll und Descent, Approach, Landing eine hohe Varianz aufweisen. Das hängt einerseits damit zusammen, dass für Take-Off und Descent, Approach, Landing häufig verschiedene Strategien angewandt werden.

Andererseits variiert das Dutch Roll Manöver dadurch stärker, dass es mit ein- und ausgeschaltetem Gierdämpfer geflogen wird. Hinzu kommt, dass Umwelteinflüsse während der Durchführung des Manövers dafür sorgen können, dass die erzeugte Roll-Gier-Schwingung verstärkt wird, wodurch ebenso die Dauer des Manövers verlängert wird. Nach dem manuellen Labeling liegt ein Datensatz mit insgesamt 11 gelabelten Klassen vor, die als natürliche Zahlen codiert werden. Dementsprechend wird dem Datensatz eine weitere Spalte für die Labels der Flugmanöver hinzugefügt, in der jeder Messpunkt, also jede Zeile, mit einem Integer-Wert zwischen 0 und 10 belegt wird. Die Codierung der Klassen, die Anzahl der gelabelten Manöver in den Daten, Informationen zur Dauer der gelabelten Manöver sowie die klassenspezifische Verteilung der Messpunkte sind in Tabelle 5-2 aufgelistet.

Tabelle 5-2: Klassenspezifische Verteilung der Flugmanöver im Rohdatensatz

Label	Manöver	Anz. Manöver	Mittlere Dauer [s]	Min. Dauer [s]	Max. Dauer [s]	Abs. Verteilung	Rel. Verteilung [%]
0	Unclassified	-	-	-	-	2.193.571	33,3
1	Take-Off	26	153,5	20,2	299,4	226.280	3,4
2	Climb	235	89,0	8,8	307,5	1.392.174	21,1
3	Descent	161	85,8	8,4	234,6	935.961	14,2
4	Turn Left	111	42,6	7,5	120,1	326.518	5,0
5	Turn Right	82	39,2	9,6	127,3	207.853	3,2
6	Dutch Roll	106	25,2	7,3	96,3	160.936	2,4
7	Elevator Sweep	33	47,0	19,5	90,4	113.219	1,7
8	Steady Pull-Up	56	45,7	34,6	89,0	193.851	2,9
9	Steady Push-Over	48	46,5	30,9	74,9	159.034	2,4
10	Descent, Approach, Landing	26	434,5	168,4	681,2	673.373	10,2

Im nächsten Schritt werden die Daten inspiziert, um sich einen genauen Überblick über Datenform, Speichergröße, Mittelwert, Median, Minima, Maxima, fehlende Werte, Datentypen und Häufigkeitsverteilung der Messwerte zu verschaffen. Da letztere sowohl für die Datenvorverarbeitung als auch den Trainingsprozess von erhöhtem Interesse ist, wird diese im Folgenden detailliert betrachtet. Anhand der signalspezifischen Häufigkeitsverteilung kann einerseits die Plausibilität der gemessenen Werte beurteilt werden und andererseits gibt sie Aufschluss über die Existenz von Ausreißern in den Daten. Ist dies der Fall, wird dadurch die Auswahl der Methode zur Standardisierung oder Normierung der Daten beeinflusst [70]. Zunächst wird die Häufigkeitsverteilung jeder Zeitreihe in einem Histogramm statistisch erfasst. Aus einer ersten Analyse der Histogramme geht hervor, dass die meisten Merkmale plausible Häufigkeitsverteilungen der Messwerte aufweisen. Auffällig sind dagegen die Messreihen der Signale ABSHUM, ALPHA, BETA, RELHUM, TD, TV, U, V, W, WA, WS, MIXRATIO, THETA_V sowie MFL_UNT, dessen Verteilungen auf Messwerte im Bereich von ca. -10.000 hinweisen (Abbildung 5-3). Dieser Eindruck bestätigt sich bei näherer Betrachtung der Boxplots von den betroffenen Messreihen. Diese sind in Abbildung 5-4 dargestellt. Die Messbereiche, in denen sich Anhäufungen von Messdaten ansammeln, sind für genannte Signale zum einen unplausibel und liegen zum anderen weit entfernt von dem Großteil der übrigen Messdaten. Daher können betroffene Messpunkte als Ausreißer bestätigt werden.

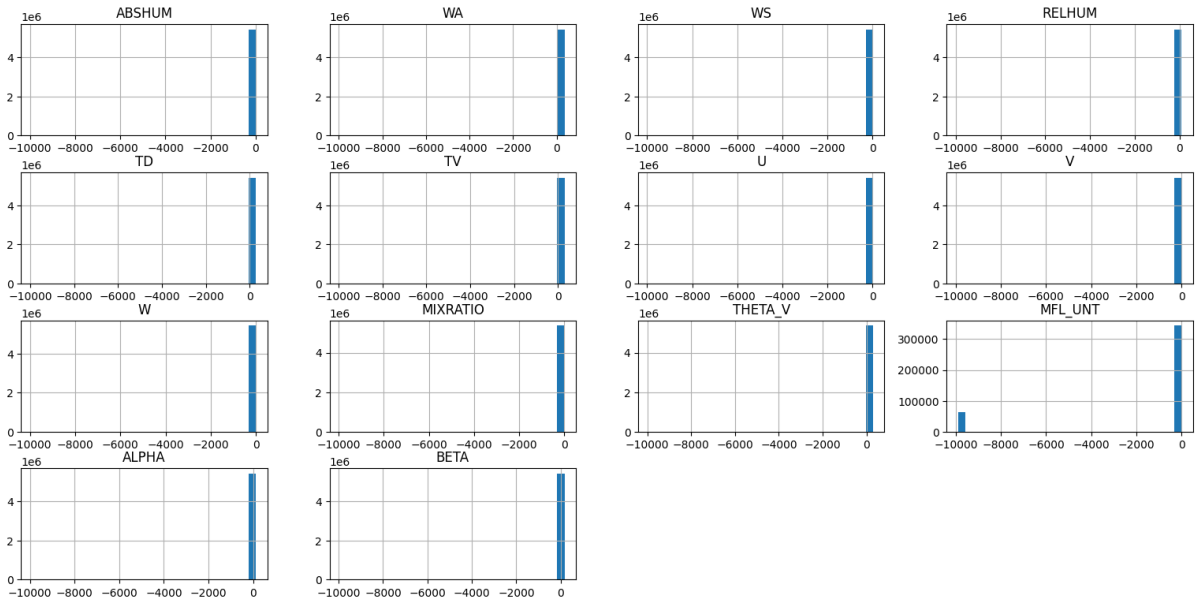


Abbildung 5-3: Ausreißerverdächtige Häufigkeitsverteilungen der Parameter Cessna Grand Cravan 208B

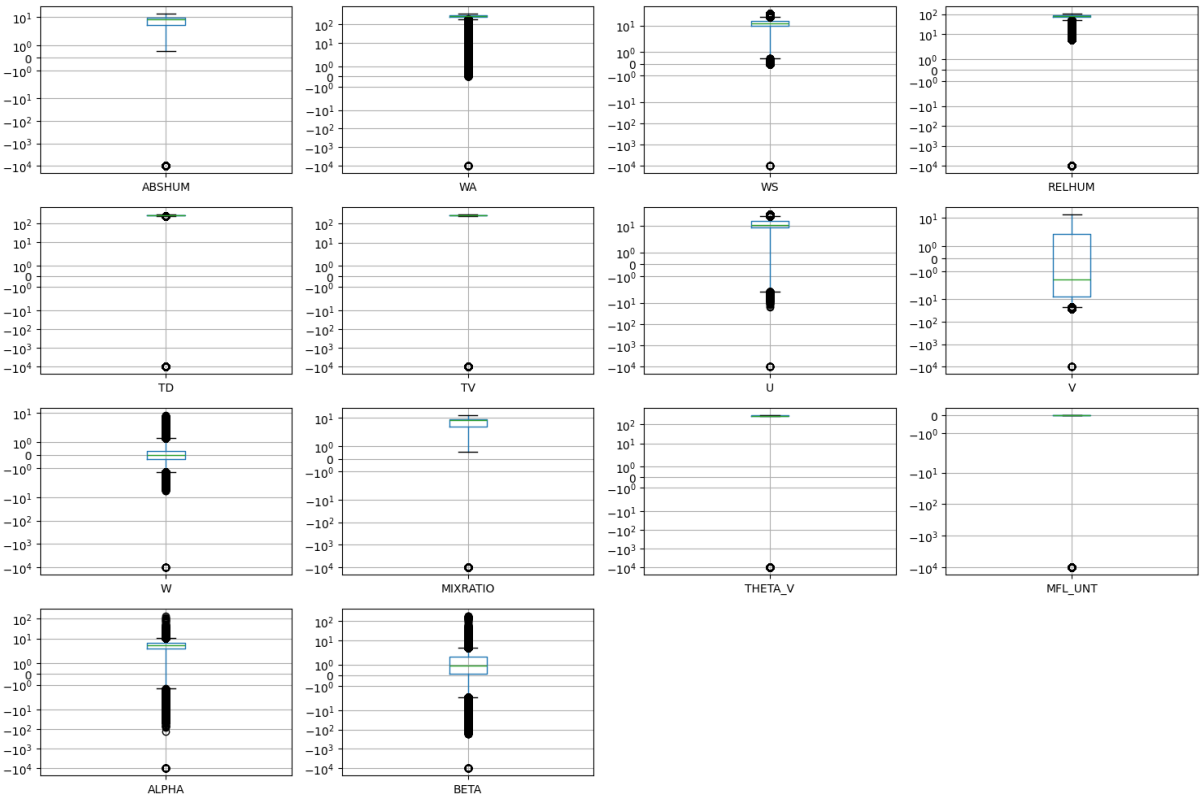


Abbildung 5-4: Boxplots der Messreihen mit Ausreißern

Nach abgeschlossener Dateninspektion werden die Flugdaten, die als Testdatensatz dienen sollen, vom Datensatz abgetrennt und separat abgelegt. Die folgenden Schritte zur Datenvorverarbeitung werden daher sowohl für Trainings- als auch Testdaten durchgeführt.

5.2 Datenvorverarbeitung

In der Phase der Datenvorverarbeitung werden die Daten mit der Python-Bibliothek Pandas zunächst in einem DataFrame-Objekt abgelegt. Darin steht eine Reihe für einen Messpunkt und die Spalten für die Sensoren bzw. Parameter und zusätzlichen Merkmale (Flugmanöver und Flugname) des jeweiligen Messpunktes. Die Datenvorverarbeitung wird für Trainings- und Testdatensatz separat durchgeführt, sodass diese für den Trainings- bzw. Testprozess ebenfalls separat geladen werden müssen. Daraus resultieren DataFrame-Objekte der Länge 5.426.670 für den Trainings- und 1.156.100 für den Testdatensatz mit jeweils 60 Spalten.

Als erstes werden die im Datensatz enthaltenen Flugnamen aus dem Datensatz entfernt. Die Flugmanöverlabels werden außerdem abgetrennt und als separate Datenreihe abgelegt. Im nächsten Schritt der Datenvorverarbeitung werden Dopplungen im Datensatz entfernt. Wie im vorherigen Abschnitt erwähnt, sind die Rohdaten mit unterschiedlichen Aufnahmefrequenzen erfasst worden. Um die ursprünglichen Daten nicht zu verfälschen, werden diese auf die

kleinste aufgenommene Datenfrequenz von 10 Hz heruntergetaktet. Wie in Kap. 5.1.3 erwähnt, fehlen für insgesamt 13 Messreihen zahlreiche Messwerte. Aufgrund der hohen Anzahl fehlender Messwerte, werden diese Messreihen nicht weiter berücksichtigt und aus dem Datensatz entfernt. Im nächsten Schritt erfolgt eine Reduktion der zu verwendenden Messsignale für das Training, um die Dimensionalität der Daten zu verringern und damit dem Problem der Überanpassung vorzubeugen. Dazu wird die Kovarianz der Messdatenparameter ermittelt. Diese gibt Auskunft darüber, ob mehrere Parameter ein ähnliches oder auffälliges Schwankungsverhalten um Ihren jeweiligen Mittelwert aufweisen. Die Kovarianz von zwei Größen kann mithilfe des Korrelationskoeffizienten (auch: Pearson-Korrelationskoeffizient) normiert und somit bewertet werden. Dieser bewegt sich in dem Intervall [-1; 1].

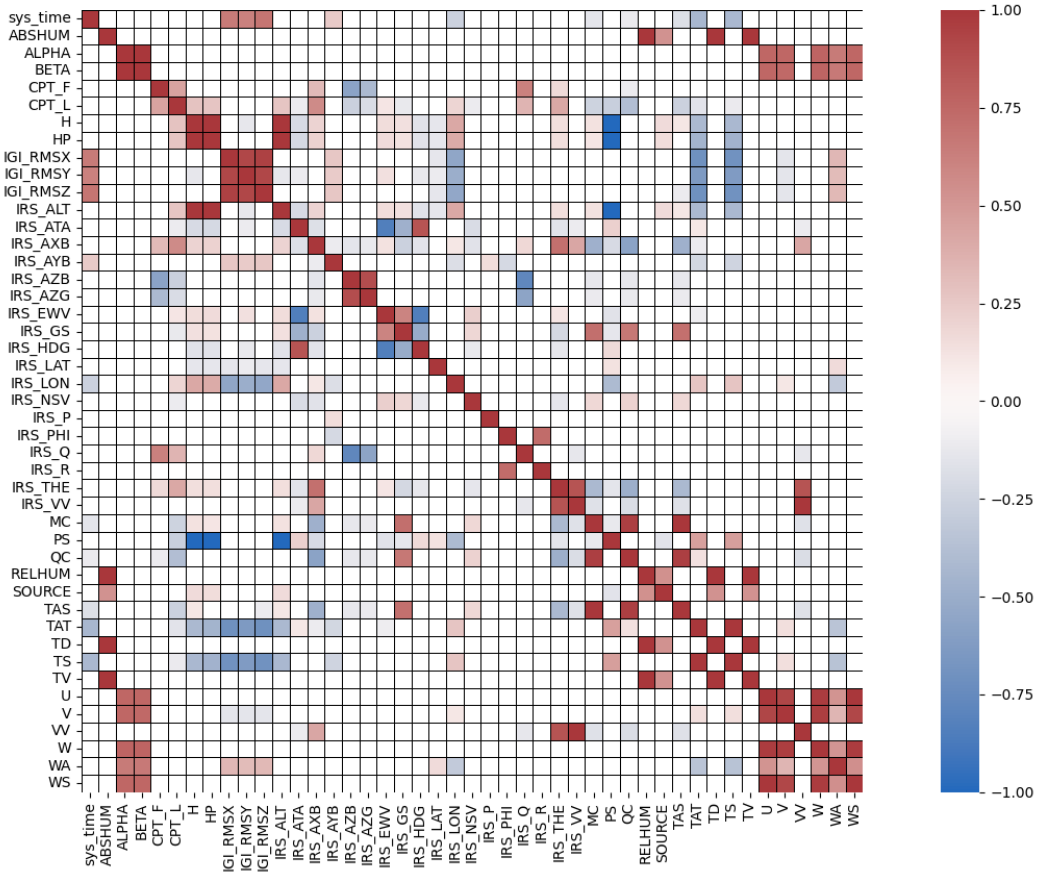


Abbildung 5-5: Korrelationsmatrix der Messdatenmerkmale

Eine Korrelation von 1 bedeutet eine maximale lineare Abhängigkeit mehrerer Merkmale. Eine Korrelation von -1 bedeutet dagegen eine maximal negative lineare Abhängigkeit mehrerer Merkmale voneinander. Im ML-Bereich wird daher zur Auswahl der relevanten Parameter häufig die Methode der Kreuzkorrelation verwendet. Dabei werden alle Datenparameter in einer Kreuztabelle (auch: Korrelationsmatrix) einander gegenübergestellt und die Kovarianzen der jeweiligen Parameterkombinationen mithilfe des Korrelationskoeffizienten berechnet.

Korrelieren Parameter stark miteinander (positiv sowie negativ), sinkt die Wahrscheinlichkeit, dass beide Merkmale zur Entscheidungsfindung eines DL-Algorithmus maßgeblich beitragen [71]. Die Korrelationsmatrix für den vorliegenden Datensatz ist in Abbildung 5-5 dargestellt. Darin ist zu erkennen, dass z. B. zwischen Anstellwinkel ALPHA sowie Schiebewinkel BETA und den Merkmalen der Windrichtung und Richtungsgeschwindigkeiten (U, V, W, WA und WS) starke Abhängigkeiten bestehen. Des Weiteren ist eine hohe negative Korrelation zwischen dem statischen Druck PS und der Flughöhe IRS_ALT, der barometrischen Flughöhe HP sowie der Höhe über dem Meeresspiegel H festzustellen. Aufgrund der beschriebenen Abhängigkeitsverhältnisse werden ALPHA, BETA und IRS_ALT stellvertretend für die beiden Merkmalsgruppen weiter berücksichtigt. In Abstimmung mit Fachexperten wie Flugversuchsingenieuren und Datenwissenschaftlern wird die Auswahl um 16 zusätzliche Merkmale erweitert. Damit werden insgesamt 19 Merkmale, die in Tabelle 5-3 aufgelistet sind, für das Training der DL-Modelle verwendet.

Tabelle 5-3: Parameterauswahl zum Training der DL-Ansätze

Lfd.-Nr.	Merkmal
1	sys_time
2	ALPHA
3	BETA
4	CPT_L
5	IRS_ALT
6	IRS_AYB
7	IRS_AZG
8	IRS_HDG
9	IRS_LAT
10	IRS_LON
11	IRS_P
12	IRS_PHI
13	IRS_Q
14	IRS_R
15	IRS_THE

16	IRS_VV
17	RELHUM
18	TAS
19	TAT

Aufgrund der zuvor erkannten hohen Ausreißerrate, werden die Daten im nächsten Schritt mithilfe der Methode der Standardisierung normalisiert [70]. Dieser Schritt ist erforderlich, damit Messreihen, dessen Wertebereiche größere Beträge aufweisen als andere, beim Training nicht stärker ins Gewicht fallen. Bei der Standardisierung der Daten werden alle Messreihen so normalisiert, dass sie einen Mittelwert von 0 und eine Standardabweichung von 1 aufweisen. Daraufhin erfolgt in der Regel eine Codierung der Labels als natürliche Ganzzahlen (auch: Integer) oder binäre Variablen. Allerdings wird dieser Schritt zunächst übersprungen, da eine Integer-Codierung der Labels bereits in dem Labeling-Prozess der Daten durchgeführt wurde (Kap. 5.1). Im darauffolgenden Schritt des Feature Engineering wird der Datensatz durch die Anwendung eines Sliding-Window grundsätzlich umstrukturiert. Für die Klassifizierung von Zeitserien ist dies bei großen Datensätzen eine gängige Methode, um den Datensatz in kürzere Datensequenzen (auch: Datensamples) aufzuteilen [72] [18] [57]. Mit einem Wert von 32 Datenpunkten pro Fenster, wird die Fenstergröße so gewählt, dass ein Fenster immer einen Datenzeitraum von 3,2 Sekunden beschreibt. Genauso wie die Fenstergröße wird die Überlappung in Anlehnung an Brownlee eingestellt [72]. Diese beträgt 50 %. Die daraus resultierenden Datensamples werden in einem 3D Numpy-Array mit dem Format [Datensamples, Datenpunkte, Datenparameter] abgelegt. Dieser Schritt muss für die Labels wiederholt werden. Durch die Verwendung eines Sliding-Window zur Sequenzierung werden Datenpunkte innerhalb einer Sequenz vereinzelt unterschiedlich gelabelt sein. Da das Neuronale Netz nach der Verarbeitung der Eingangssequenz allerdings nur eine Vorhersage über die gesamte Sequenz ausgibt, muss für jede Sequenz ein Label bestimmt werden. Eine Methode für die Klassifizierung von Datensequenzen ist dabei die Verwendung des letzten Labels einer Sequenz [73]. Daraus geht entsprechend ein 1D Numpy-Array hervor, das für jedes Datensample ein Integer-codiertes Label enthält. Datensamples und Label werden anschließend an die Pipeline zum Modelltraining der Modelle übergeben. Außerdem werden die Daten jeweils in einer npy-Datei gespeichert und auf dem Datenlaufwerk der Arbeitsgruppe Flight Test Instrumentation der Einrichtung Flugexperimente abgelegt. So kann bei Anpassungen der Methoden zur Vorverarbeitung auf die Daten erneut zugegriffen werden.

Eine grobe Übersicht der Operationen, die während der Datenvorverarbeitung mithilfe des Python-Codes (digitaler Anhang) ausgeführt werden, ist dem Anhang A 2 zu entnehmen.

5.3 Modelltraining

Die Phase für das Modelltraining wird in dieser Arbeit grundlegend in drei Phasen unterteilt: die Datenvorbereitung, die Trainingsdurchführung und eine erste Auswertung der Leistungsfähigkeit der Modelle. Zu Beginn muss der verwendete Datensatz für das Training vorbereitet und in das benötigte Datenformat konvertiert werden. Darin inbegriffen ist die Aufteilung der Daten in Trainings-, Validierungs- und Testdatensatz. Je nach Deep Learning Ansatz und verwendeten Bibliotheken zum Training der Modelle, werden zudem unterschiedliche Anforderungen an das Datenformat der Eingangsdaten gestellt. Diese müssen in der Datenvorbereitung berücksichtigt werden, weshalb spätestens zu diesem Zeitpunkt die zu verwendenden Netzarchitekturen und Bibliotheken definiert werden müssen. Die Konfiguration sowie das Training der Modelle erfolgt in dieser Arbeit mit der Bibliothek Pytorch und dem darauf aufbauenden Framework Pytorch Lightning. Die Trainingsdurchführung umfasst das Training sowie die Validierung der definierten Netzarchitekturen. Um die Leistungsfähigkeit der Modelle auszuwerten, werden die Modelle anhand des Testdatensatzes verifiziert, die das Modell im Trainingsprozess nicht verwendet. Anschließend werden die Architekturen miteinander verglichen. Die Datenvorbereitung und die Durchführung des Trainings werden im Folgenden für einen ResNet-, einen LSTM- sowie einen MLP-Ansatz durchgeführt. Im Anschluss daran erfolgt die Auswertung und der Vergleich der Ergebnisse aller drei Ansätze.

Das ResNet-Modell konnte in einem umfassenden Benchmark mit verschiedenen CNN-Architekturen zur Klassifizierung unterschiedlicher Zeitserien im Durchschnitt die besten Ergebnisse erzielen [16]. Dabei kam heraus, dass ResNet zudem bei der Klassifizierung längerer Zeitserien für die meisten Datensätze (75 %) die besten Ergebnisse erzielt. Außerdem schnitt ResNet beim Training mit größeren Trainingsdatensätzen im Vergleich zu den anderen untersuchten Architekturen am besten ab. Darüber hinaus konnte beobachtet werden, dass sich die Performance von ResNet mit wachsender Anzahl der Trainingsdaten immer stärker von den Wettbewerbern abhob.

Als zweites Vergleichsmodell wurde ein LSTM-Ansatz ausgewählt, da sich dieser Ansatz unter anderem bereits als erfolgreiche Methode für die Klassifizierung und Vorhersage von Flugzeugtrajektorien, Flugmanövern und Flugphasen erwiesen hat [63] [18] [74].

Als dritter Ansatz wird ein MLP ausgewählt, das mit physikalischen binären Merkmalsvektoren aus den erstellten Datensequenzen gefüttert wird. Einen ähnlichen Ansatz verfolgen Tchuente et al. zur Erkennung von abrupten menschlichen Bewegungen auf Basis von Handydaten [22]. Dieser Ansatz zielt vor allem auf die Reduktion der Dimensionen ab, um Overfitting und das Konvergieren in lokalen Minima der Fehlerfunktionen möglichst zu vermeiden.

Aufgrund der verbreiteten erfolgreichen Anwendung von 1D-CNNs und LSTMs zur Klassifizierung multivariater Zeitseriendaten in Kombination sowie im Einzelnen, wird erwartet, dass sich die Ansätze ResNet und LSTM auch im Bereich der Manöverklassifizierung von dem MLP-Ansatz abheben.

5.3.1 ResNet

Vorbereitend für das Training wird der Trainingsdatensatz zuerst in einen Trainings- und Validierungsdatensatz unterteilt. Diese Aufteilung dient dazu, dass während des Trainings die Validierungsdaten dazu verwendet werden können, die Leistungsfähigkeit des Modells zu überwachen und basierend auf den Beobachtungen Verbesserungsmaßnahmen am Modell selbst oder in der Datenvorverarbeitung vorzunehmen. Dazu wird ein Verhältnis von 80:20 der Trainings- und Validierungsdaten gewählt, sodass die Validierungsdaten insgesamt einen Anteil von 16 % des gesamten Datensatzes ausmachen (Abbildung 5-6). Damit umfassen die Trainingsdaten schließlich 66 % des gesamten Datensatzes.

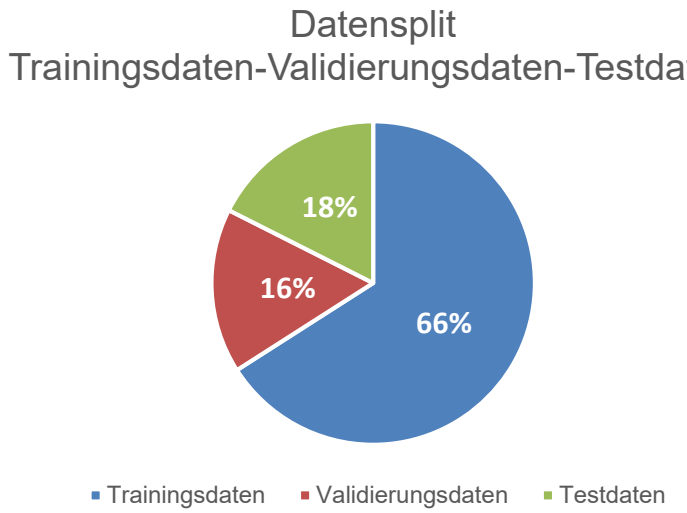


Abbildung 5-6: Datensplit Trainingsdaten-Validierungsdaten-Testdaten

Die Faltung der Daten wird in Python mit der Klasse Conv1d aus dem torch.nn Modul durchgeführt. Dafür müssen die zweite und dritte Dimension des 3D-Numpy Arrays getauscht werden, sodass die Eingangsdaten das dafür erforderliche Format [Datensamples,

Datenparameter, Zeitschritte] erhalten. Die Konfiguration der ResNet-Architektur orientiert sich an der Struktur von Fawaz et al. und besteht daher aus drei Faltungsblöcken mit jeweils drei Faltungsschichten, gefolgt von einer Global Average Pooling Schicht sowie einem Softmax-Klassierer als letzte Schicht (Tabelle 5-4) [16].

Tabelle 5-4: Modellhyperparameter ResNet nach Fawaz et al. [16]

Schicht	Filteranzahl	Ausgabegröße	Anzahl trainier-bare Parameter
Eingangsschicht	-	(Batch Größe, 32, 19)	-
Conv1D(f=8, s=1)	64	(Batch Größe, 32, 64)	9.728
Conv1D(f=5, s=1)	64	(Batch Größe, 32, 64)	20.480
Conv1D(f=3, s=1)	64	(Batch Größe, 32, 64)	12.288
„Skip Connection“: Conv1D(f=1, s=1)	64	(Batch Größe, 32, 64)	1.216
Conv1D(f=8, s=1)	128	(Batch Größe, 32, 128)	65.536
Conv1D(f=5, s=1)	128	(Batch Größe, 32, 128)	81.920
Conv1D(f=3, s=1)	128	(Batch Größe, 32, 128)	49.152
„Skip Connection“: Conv1D(f=1, s=1)	128	(Batch Größe, 32, 128)	8.192
Conv1D(f=8, s=1)	128	(Batch Größe, 32, 128)	131.072
Conv1D(f=5, s=1)	128	(Batch Größe, 32, 128)	81.920
Conv1D(f=3, s=1)	128	(Batch Größe, 32, 128)	49.152
„Skip Connection“	-	(Batch Größe, 32, 128)	-
AvgPool(s=1)	-	(Batch Größe, 128, 1)	-
Softmax	-	(Batch Größe, 11, 1)	1.419

Die Anzahl der Filter pro Faltungsschicht bleibt innerhalb der Faltungsblöcke konstant. Im ersten Block werden 64 Filter angewandt, im zweiten Block 128 und im dritten und letzten Block ebenfalls 128. Die Filter der Schichten innerhalb der Blöcke besitzen eine Länge l von 8, 5 und 3 Zeitschritten und tasten die Daten mit einer Schrittweite s von $s = 1$ ab. Auf jede

Datensequenz wird ein „Zero Padding“ in der notwendigen Größe angewandt, sodass die Länge der Datensequenz über die verschiedenen Faltungsschichten konstant bleibt. Zudem wird nach jeder Faltung eine Batch-Normalisierung durchgeführt, bevor die ReLU-Aktivierungsfunktion auf das Ergebnis angewandt wird. Nach den Faltungsblöcken berechnet eine Global-Average-Pooling-Schicht den Durchschnitt der gefilterten Merkmale, reduziert damit die Dimensionalität der Daten und soll so Overfitting vermeiden. Der Softmax-Klassierer wird zum Schluss verwendet, um eine Klassifizierung des Manövers in dem betrachteten Zeitfenster durchzuführen.

Für das Training werden die Trainingsdatensequenzen vorab in sogenannten Mini-Batches (dt.: Mini-Stapel) geladen. ResNet wird mit diesen Stichproben der Datensequenzen gefüttert und berechnet für jede Datensequenz eines Mini-Batches eine Wahrscheinlichkeitsvorhersage für die Klassen. Diese Vorhersage wird über die Verlustfunktion mit den wahren Labels abgeglichen und zur Berechnung des Gradienten für jede Datensequenz verwendet. Um die Gewichte anzupassen wird in diesem Fall der durchschnittliche Gradient eines Mini-Batches berechnet und angewandt. Bevor die Datensequenzen in die Mini-Batches geladen werden, wird dessen Reihenfolge randomisiert, damit das ResNet keine spezielle Reihenfolge der Datensequenzen erlernt. Für das Training von ResNet wird in Anlehnung an Fawaz et al. eine Batchgröße von 16 verwendet. Als Verlustfunktion wird die Cross-Entropy verwendet, da diese Funktion als ein Industriestandard für die Fehlerfunktion bei Mehrklassenklassifizierungen durch Neuronale Netze gilt [43]. Für die Optimierung der Gewichte wird der im Bereich des Deep Learning weit verbreitete Adam-Algorithmus verwendet [44]. Das Training wird über 50 Epochen durchgeführt, sodass die ResNet-Struktur 50 Mal mit allen Trainingsdatensequenzen gefüttert wird, bevor es fertig trainiert ist. Der Adam-Optimierer wird in Anlehnung an Ismail Fawaz et al. und Oppermann mit 0,001 initialisiert [16] [75]. Die Trainingshyperparameter für das Training von ResNet werden in Tabelle 5-5 zusammengefasst.

Tabelle 5-5: Trainingshyperparameter ResNet

Hyperparameter	Wert
Batchgröße	16
Fehlerfunktion	Cross-Entropy
Optimierungsalgorithmus	Adam
Anzahl max. Epochen	50
Lernrate	0,001

5.3.2 LSTM

Zur Vorbereitung des Datensatzes für das Training des LSTM-Modells wird die Aufteilung in Trainings- und Validierungsdatensätze von der Datenvorbereitung für das ResNet-Modell adaptiert. Da die Klasse LSTM aus dem torch.nn-Modul die Eingangsdaten in demselben Format [Datensamples, Zeitschritte, Datenparameter] erwartet, in dem der Datensatz bereits vorliegt, muss an dieser Stelle keine weitere Transformation der Daten erfolgen.

Die Struktur des LSTM-Netzes orientiert sich an der LSTM-Struktur in der hybriden CNN-LSTM Architektur von Ma und Tian (Tabelle 5-6) [18]. Deren Modell wurde zur Vorhersage von Flugzeugtrajektorien entwickelt und eignet sich daher für den vorliegenden Anwendungsfall der Manöverdetektion. In insgesamt vier Schichten werden nach der Eingangsschicht zwei LSTM-Schichten mit jeweils 50 Einheiten hintereinandergeschaltet. Außerdem wird nach jeder LSTM-Schicht Dropout mit 20 % Abschaltrate angewandt. Die Klassifizierung der Ausgangsdaten erfolgt in einer vollvernetzten Schicht.

Tabelle 5-6: Modellhyperparameter LSTM nach Ma und Tian [18]

Schicht	Ausgabegröße	Anzahl trainierbare Parameter
Eingangsdatensequenz	(Batch Größe, 32, 19)	-
LSTM	(Batch Größe, 32, 50)	14.200
LSTM	(Batch Größe, 50, 50)	20.400
Softmax	(Batch Größe, 11)	561

Für das Training werden zur besseren Vergleichbarkeit die Trainingshyperparameter des zuvor beschriebenen ResNet-Ansatzes übernommen. Diese sind in Tabelle 5-5 aufgeführt.

5.3.3 MLP

Für den MLP-Ansatz nimmt die Vorbereitung des Datensatzes verglichen zu dem ResNet- und LSTM-Ansatz mehr Aufwand in Anspruch. In diesem Fall, werden die Merkmale in den sequenzierten Daten zur Unterscheidung der gelabelten Manöver während der Datenvorverarbeitung manuell entwickelt und als binäre Merkmalsvektoren an ein MLP übergeben. Zu diesem Zweck werden die Datensequenzen jeweils auf 10 verschiedene physikalische Merkmale überprüft (Tabelle 5-7). Die Merkmale werden nach erneuter Analyse der Messsignale verschiedener Sensoren ausgewählt und zum Ende der

Datenvorverarbeitung implementiert. Zum einen werden die Datensequenzen darauf überprüft, ob die aktuelle Flughöhe über Grund $AGL < 121{,}92\ \mathrm{m} = 400\ ft$ ist. Dadurch soll das Manöver Take-Off eindeutig identifiziert werden. Der Grenzwert wird durch das erste Austrimmen eines Flugzeugs bei 400 ft während der Abflugphase bestimmt. Anhand dessen, ob die Steiggeschwindigkeit v_z innerhalb einer Sequenz stets größer oder kleiner null ist, soll identifiziert werden, ob ein Steig- oder Sinkflug vorliegt. Gleiches gilt für die Überwachung des Rollneigungswinkels θ, um eine geflogene Links- oder Rechtskurve zu identifizieren. Dafür wird ebenfalls geprüft, ob die Änderungsrate des Gierwinkels r innerhalb einer Sequenz stets positiv oder negativ ist. Zur Identifizierung eines Dutch Roll Manövers soll untersucht werden, ob für die Spannweite R – also die Differenz zwischen Minimal- und Maximalwert – der y-Beschleunigung $R_{a_y} > 3\frac{m}{s^2}$ innerhalb einer Sequenz gilt. Zur Identifizierung vom Steady Pull-Up und Steady Push-Over Manöver wird überprüft, ob für die Spannweite der Steiggeschwindigkeit $R_{v_z} > 30\frac{m}{s}$ gilt. Für das Elevator Sweep Manöver wird geprüft, ob für die Datenspannweite der Steuerhornstellung $R_{CPT_L} > 10\ mm$ gilt. Bei der erneuten Analyse der Flugdaten haben sich diese Datenspannweiten als charakteristisches quantitatives Merkmal der jeweiligen Manöver herausgestellt, weswegen sie an dieser Stelle als Eingangsmerkmale des MLP dienen sollen. Um zusätzlich das Manöver Descent, Approach, Landing zu identifizieren, wird erfasst, ob sich die Eingangsdaten innerhalb der letzten 7 Minuten und 14 Sekunden des Fluges befinden. Der Wert orientiert sich an der durchschnittlichen Dauer der aufgenommenen Manöver Descent, Approach, Landing aus den Trainingsdaten. Der daraus resultierende binäre Merkmalsvektor jeder Datensequenz mit der Länge 11 wird anschließend in einem 2D Numpy-Array mit dem Format [Datensamples, Physikalische Merkmale] abgelegt und an die Pipeline für das Modelltraining übergeben. Der Datensatz wird, wie zuvor für ResNet und LSTM beschrieben, aufgeteilt. Da die Linear-Klasse des torch.nn-Moduls die Daten im oben beschriebenen Format erwartet ist außerdem keine Umstrukturierung der Daten notwendig. Die Dimensionalität der Eingangsdaten wurde bei diesem Ansatz deutlich reduziert und damit auch die Komplexität. Um aus den binären Eingangsdaten komplexe Zusammenhänge zur Detektion der Manöver abzuleiten, wird mit drei verdeckten Schichten die Netzstruktur vertieft [76]. Die Anzahl der Neuronen pro Schicht wird außerdem von Schicht zu Schicht erhöht. Damit sollen die als binäre Merkmalsvektoren codierten Flugdaten wieder decodiert werden. Daher sind die verdeckten Schichten mit 16, 32 und 64 Neuronen ausgestattet. Aufgrund der weiten Verbreitung in anderen DNNs wird ReLU als Aktivierungsfunktion für alle drei verdeckten Schichten verwendet [76]. Die Klassifizierung in der letzten Schicht erfolgt erneut durch einen Softmax-Klassierer. Die Modellhyperparameter sind in

Tabelle 5-8 zusammengefasst.

Tabelle 5-7: Binäre Merkmalsüberprüfung von Zeitsequenzen

Lfd.-Nr.	Parameter P	Merkmal innerhalb der Datensequenz eines Parameters x_P
1	AGL	$\forall x \in x_{AGL}, x < 121{,}9\ m$
2	v_z	$\forall x \in x_{v_z}, x > 0$
3	v_z	$\forall x \in x_{v_z}, x < 0$
4	Φ	$\forall x \in x_{\Phi}, x > 0$
5	Φ	$\forall x \in x_{\Phi}, x < 0$
6	r	$\forall x \in x_{r}, x > 0$
7	r	$\forall x \in x_{r}, x < 0$
8	a_y	$\max\left(x_{a_y}\right) - \min\left(x_{a_y}\right) > 3\frac{m}{s^2}$
9	v_z	$\max\left(x_{v_z}\right) - \min\left(x_{v_z}\right) > 30\frac{m}{s}$
10	CPT_L	$\max\left(x_{CPT_L}\right) - \min\left(x_{CPT_L}\right) > 10\ mm$
11	sys_time	$\forall x \in x_{sys\_time}, x > 434\ s$

Tabelle 5-8: Modellhyperparameter MLP

Schicht	Ausgabegröße	Anzahl trainierbarer Parameter
Eingang Merkmalsvektoren	(Batch Größe, 11)	-
Dense	(Batch Größe, 16)	192
Dense	(Batch Größe, 32)	544
Dense	(Batch Größe, 64)	2.112
Softmax	(Batch Größe, 11)	715

Zur besseren Vergleichbarkeit werden die Trainingshyperparameter von den zuvor beschriebenen Ansätzen ebenfalls übernommen. Diese sind in Tabelle 5-5 aufgeführt.

5.3.4 Zwischenergebnisse

Jeder Ansatz wurde mit jeweils drei Durchläufen trainiert. Dabei wurden die Trainingsdatensequenzen nach dem ersten und zweiten Durchlauf aller Modelle neu durchmischt, um die Modelle auf unterschiedlichen Untergruppierungen der Daten bewerten zu können. Die Metriken wurden in Tensorboard visualisiert, wo die Scores jeweils über den Trainingsschritten aufgetragen wurden. Ein Trainingsschritt ist hierbei gleichbedeutend mit einer Aktualisierung der Gewichte. Die Fehlerkurven auf den Trainings- und Validierungsdaten sind in Abbildung 5-7 zu erkennen. Zudem zeigt Abbildung 5-8 den Trainings- und Validierungsscore vom Recall aller Modelle. Abbildung 5-9, Abbildung 5-10 und Abbildung 5-11 zeigen die Wahrheitsmatrizen der besten Validierungsergebnisse aller drei Ansätze und Abbildung 5-12 zeigt außerdem die MCC-Verläufe für Training und Validierung. Die Validierungsergebnisse für den Fehler, Recall und MCC aller drei Ansätze sind zudem in Tabelle 5-9 aufgelistet. Dabei wird zwischen dem Mittelwert $\bar{x}$ und der Standardabweichung σ unterschieden.

Tabelle 5-9: Validierungsergebnisse ResNet, LSTM, MLP

Metrik	ResNet		LSTM		MLP	
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
Fehler	1,726	0,0026	**0,272**	**0,0034**	2,211	0,0054
Recall	0,651	0,0043	**0,895**	0,005	0,091	**0**
MCC	0,7722	0,0031	**0,9022**	0,0014	0	**0**

Allgemein ist zu beobachten, dass alle drei Trainingsdurchläufe der unterschiedlichen Ansätze jeweils ähnliche Ergebnisse aufweisen. Daher kann davon ausgegangen werden, dass diese aussagekräftig und reproduzierbar sind. Anhand der Trainings- und Validierungsverläufe ist zu erkennen, dass der LSTM-Ansatz (grün) im Vergleich zum ResNet- (gelb) und MLP-Ansatz (rot) die besten Ergebnisse im Training liefert. Weiterhin kann eine deutliche Leistungsdifferenz zu dem MLP-Ansatz festgestellt werden, was ein erster Hinweis darauf ist, dass das MLP-Modell unterangepasst ist. Fehler und Recall zeigen zusätzlich, dass das MLP-Modell bereits sehr früh nach ca. 5000 Trainingsschritten konvergiert.

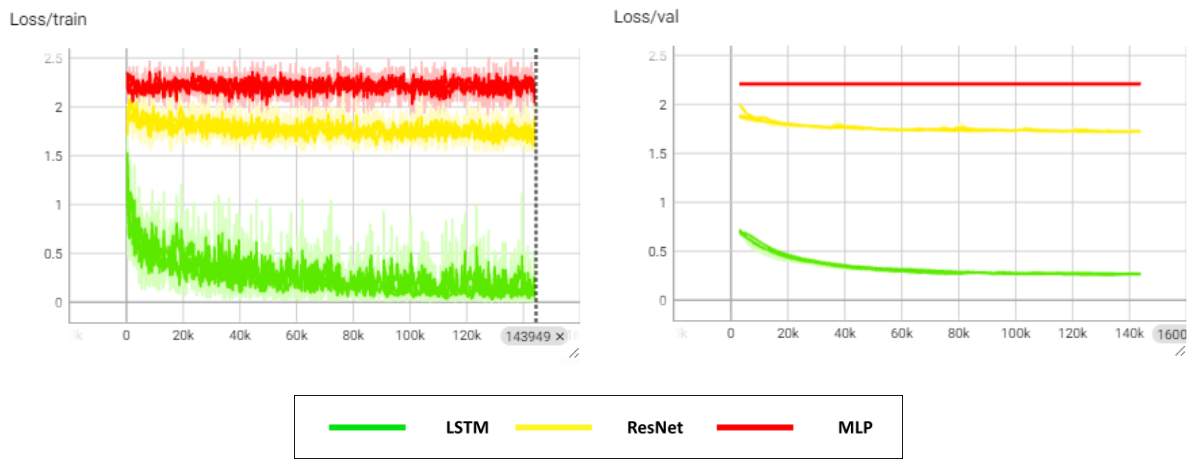


Abbildung 5-7: Vergleich Fehlerverläufe Training-Validierung

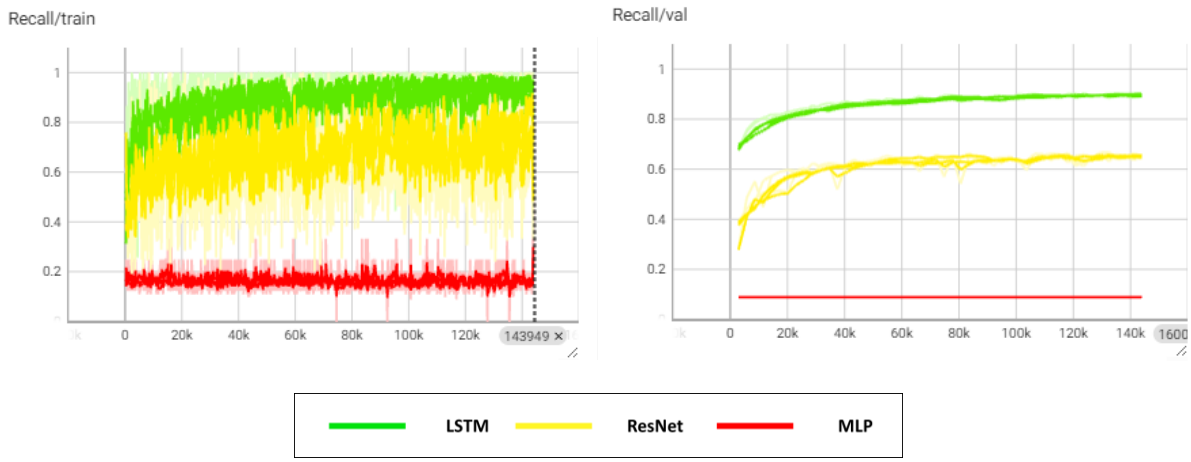


Abbildung 5-8: Vergleich Recall Scores Training-Validierung

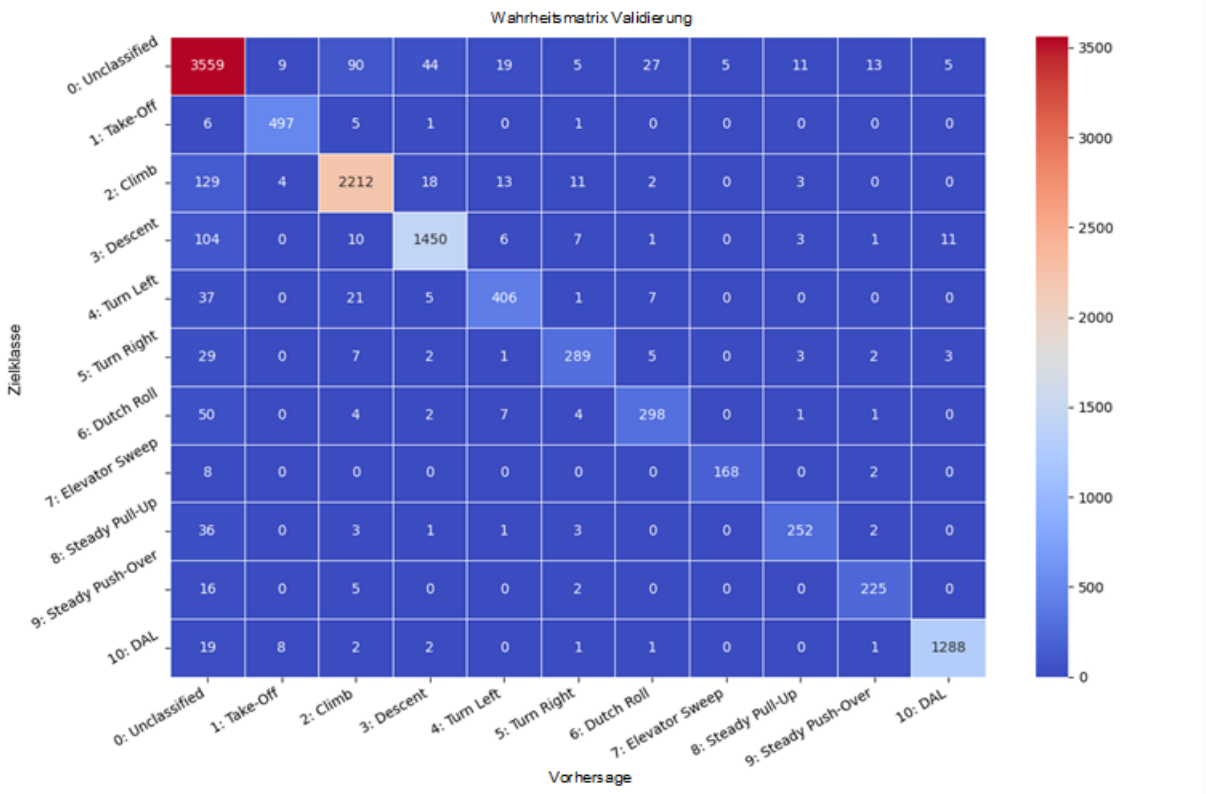


Abbildung 5-9: Wahrheitsmatrix Validierung LSTM

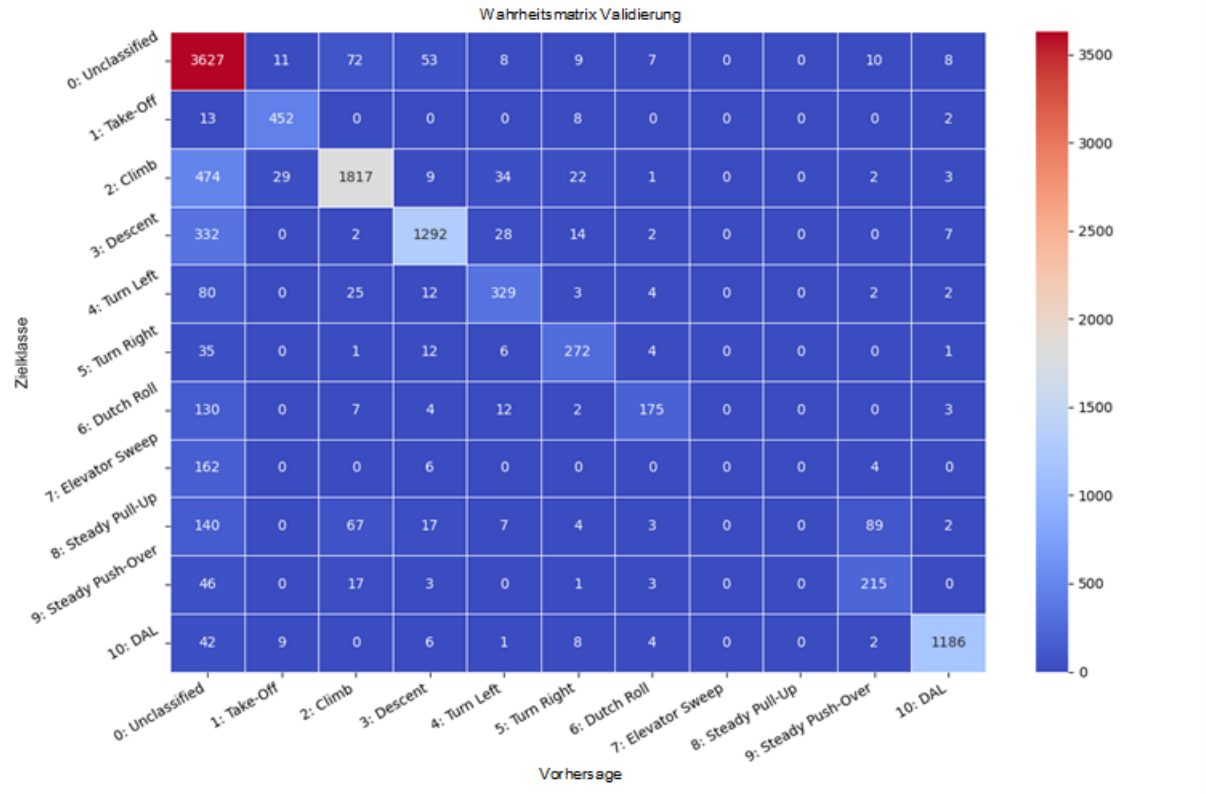


Abbildung 5-10: Wahrheitsmatrix Validierung ResNet

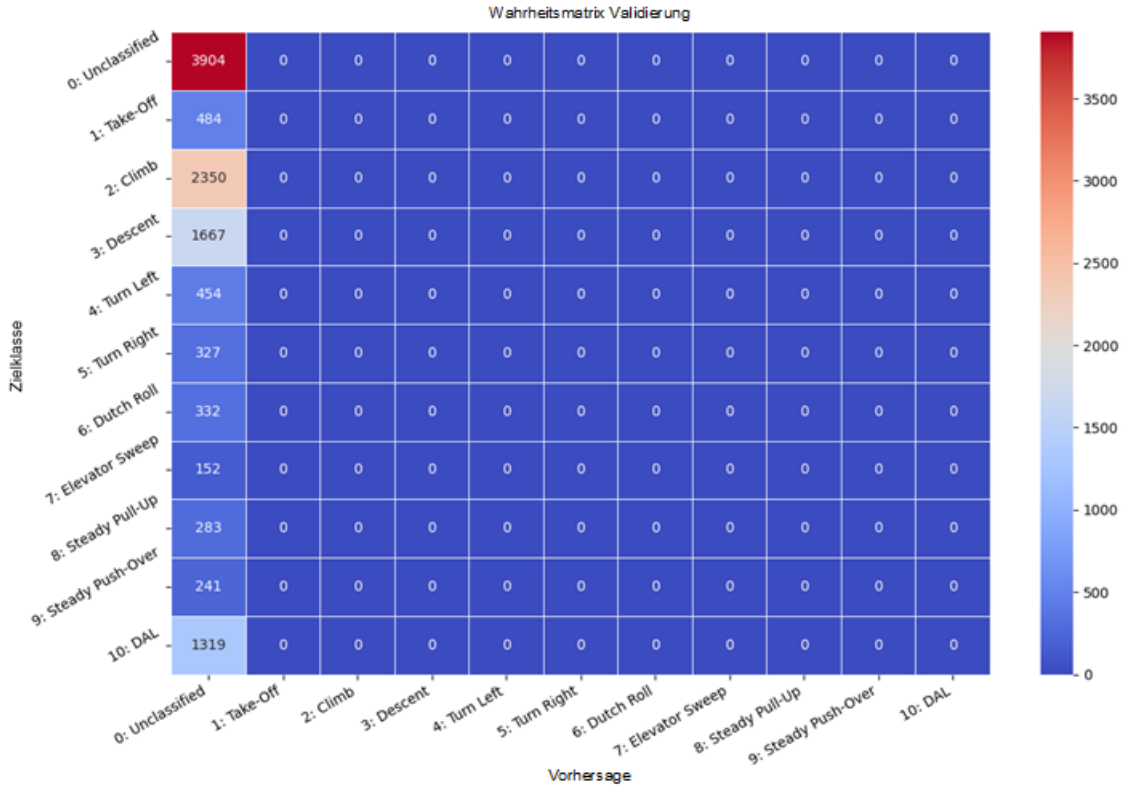


Abbildung 5-11: Wahrheitsmatrix Validierung MLP

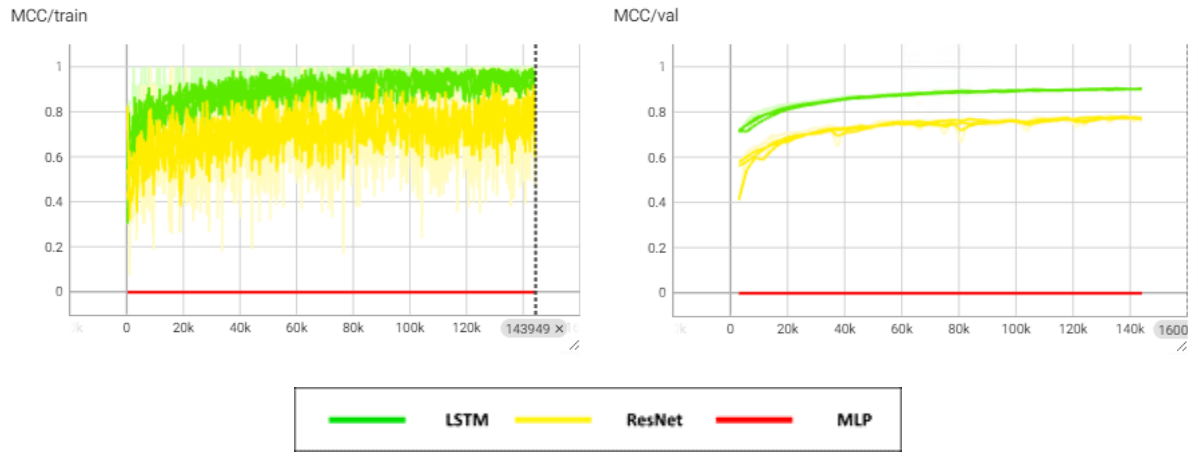


Abbildung 5-12: Vergleich MCC-Scores Training-Validierung

Die Mehrklassen-Wahrheitsmatrix gibt Auskunft darüber wie ein Modell die verschiedenen Dateninstanzen klassifiziert hat und zeigt, welcher Klasse sie eigentlich zugehörig sind. Die Wahrheitsmatrix eines perfekten Modells würde dementsprechend nur die Hauptdiagonale ausfüllen. In diesem Fall wird der MCC als zusätzliche Leistungsmetrik betrachtet, um die Inhalte der Wahrheitsmatrix in einem Wert quantitativ zu verdeutlichen. Anhand der Wahrheitsmatrizen lässt sich ablesen, dass der LSTM-Ansatz über alle Klassen hinweg die besten Ergebnisse liefert. Auffällig ist, dass das Dutch Roll Manöver dabei schlechter erkannt

wird, was sich auch in dem klassenspezifischen Recall wiederspiegelt (71,5 %). Ursächlich kann dafür die erhöhte Variabilität sein, mit der die Manöver durchgeführt werden und so auch in den Daten vertreten sind. Die Wahrheitsmatrix des besten ResNet-Modells zeigt, dass überraschenderweise zwei der vier komplexen Flugmanöver auf den Validierungsdaten gar nicht erkannt werden können. Dazu gehören das Elevator Sweep und das Steady Pull-Up Manöver. Bemerkenswert dabei ist, dass nahezu alle Elevator Sweep Datensamples als Unclassified vorhergesagt werden. Zudem wird das Steady Pull-Up Manöver häufig fälschlicherweise als Climb oder Steady Push-Over Manöver identifiziert. Kurze Steigflugphasen kommen ebenfalls während des Steady Pull-Up vor. Daher könnte die Fenstergröße des Sliding-Window für ResNet möglicherweise nicht dafür geeignet sein, um einen ausreichend großen räumlichen Kontext der Daten des Flugmanövers zu erfassen. Die häufige Verwechslung von Steady Push-Over und Steady Pull-Up Manöver, könnte darauf ebenfalls hinweisen. Die Wahrheitsmatrix für das MLP-Modell bestätigt den zuvor gewonnenen Eindruck und zeigt zudem, dass das Modell keines der gelabelten Flugmanöver erkennt. Das deutet weiter darauf hin, dass sich der MLP-Ansatz in der gewählten Form für die automatisierte Erkennung von Flugmanövern nicht eignet.

Eine grobe Übersicht der Operationen, die während des Modelltrainings mithilfe des Python-Codes (digitaler Anhang) ausgeführt werden, ist dem Anhang A 2 zu entnehmen. Zusätzlich zeigt Anhang A 3 beispielhaft die Konfiguration für den Python-Code, mit der die Datenvorverarbeitung sowie das Training selbst der ersten LSTM-Modelle durchgeführt wurde. Je nach Modell wurden dabei die verschiedenen Einstellungsmöglichkeiten entsprechend der Kommentare im Code für das Training des jeweiligen Modells angepasst.

5.4 Hyperparametersuche und Test

Um abzusichern, dass gute oder schlechte Ergebnisse der Modelle nicht auf Grundlage von zufällig gut oder schlecht ausgewählten Hyperparametern resultieren, wird nach den ersten Trainingsdurchläufen der Modelle eine Hyperparametersuche durchgeführt. Dafür wird die Python-Bibliothek Optuna verwendet, in der verschiedene Suchalgorithmen zur Verfügung stehen, darunter auch die Zufallssuche [77].

Zur Anpassung der Hyperparameter kann der Prozess der Parametersuche manuell durchgeführt werden. Da diese Vorgehensweise jedoch meist zeitaufwändig und fehleranfällig ist, sind stattdessen die Ansätze Grid Search (dt.: Rastersuche) und Random Search (dt. Zufallssuche) für die Anpassung der Hyperparameter weiter verbreitet [78] [79] [80]. Bei der Rastersuche wird ein Set von zu verändernden Hyperparametern ausgewählt und eine Menge möglicher Parameterausprägungen definiert. Anschließend werden Modelle mit allen

möglichen Parameterkonfigurationen trainiert und anhand der Ergebnisse das beste Modell ausgewählt. Da hierbei der Zeit- und Rechenaufwand schon bei geringer Anzahl von Parametern und Parameterausprägungen stark ansteigt, wurde der Ansatz der Zufallssuche entwickelt. Anstatt diskrete Parameterausprägungen zu bestimmen, werden dabei lediglich Grenzen für die verschiedenen Hyperparameter definiert. Anschließend wird eine bestimmte Anzahl von Konfigurationen festgelegt, für welche die Hyperparameterwerte innerhalb dieser Grenzen zufällig ausgewählt werden. Vorteilhaft ist bei dieser Methodik zudem, dass der Suchraum nicht auf diskrete Werte beschränkt ist und die Hyperparameter stattdessen innerhalb der definierten Grenzwerte frei bestimmt werden können. Sowohl Raster- als auch Zufallssuche sind im Bereich der Flugdatenverarbeitung bereits erfolgreich angewandt worden [63] [74] [81]. Im Fall des LSTM- und MLP-Ansatzes werden Batchgröße, Lernrate, Anzahl der LSTM-Neuronen und Anzahl der Schichten variiert. Bei der Anpassung der Parameter von ResNet werden anstelle der Anzahl der Neuronen pro Schicht, die Anzahl der angewandten Filter pro „Residual Block“ variiert. Die Anzahl der Schichten wird nicht variiert. Tabelle 5-10 fasst zusammen, innerhalb welcher Grenzen die oben beschriebenen Hyperparameter für die jeweiligen Ansätze zufällig ausgewählt werden. In runden Klammern ist jeweils die Referenz der Grenzen angegeben. Die Auswahl der Grenzen orientiert sich dabei an Rakhshani et al. [82], Lara-Benitez [83], Youness et al. [84] und Marco et al. [85], welche die Raster- und Zufallssuche für ResNet-, LSTM- und MLP-Modelle bereits erfolgreich anwenden konnten.

Tabelle 5-10: Intervalle Hyperparameter für Zufallssuche

Ansatz / Parameter	ResNet	LSTM	MLP
Batchgröße	[8, 256]; ([82])	[64, 256]; ([83])	[8, 256]; ([82])
Lernrate	[0,0001, 0,9]; ([82])	[0,0001, 0,1]; ([83])	[0,001, 0,01]; ([84])
Anzahl der Schichten	-	[1, 3]; ([85])	[2, 3]
Anzahl der Neuronen	-	[32, 128]; ([83])	[4, 144]; ([84])
Anzahl der Filter	[8, 256]; ([82])	-	-

Zur Parametersuche wird jeder Ansatz mit jeweils 50 zufällig ausgewählten Konfigurationen über jeweils 50 Epochen trainiert. Zusätzlich wird Early Stopping angewandt, um die Trainingszeit zu reduzieren und Overfitting möglichst zu vermeiden. Verringert sich der beste Fehlerscore in den Validierungsdaten innerhalb von fünf Epochen nicht mehr um min. 0,001, wird somit das Training einer Modellkonfiguration gestoppt. Hinsichtlich der erwarteten Ergebnisse werden drei Hypothesen H1-H3 aufgestellt, die nach durchgeführter Parametersuche verifiziert oder falsifiziert werden.

H1: Eine hohe Batchgröße und hohe Lernrate wird sich positiv auf den Recall von ResNet auswirken

Die initialen Trainingshyperparameter wurden für alle Modelle mit einer relativ kleinen Batchgröße von 16 eingestellt. Allerdings konnte für ResNet-Modelle in der Bildklassifizierung bereits festgestellt werden, dass eine hohe Batchgröße zu besseren Ergebnissen führen kann und die Generalisierungsfähigkeit eines Modells dadurch nicht negativ beeinflusst wird [86]. Daher wird die Hypothese aufgestellt, dass in diesem Fall eine Batchgröße im Bereich von 128 bis 256 zu besseren Ergebnissen des ResNet-Ansatzes führen wird.

Zudem besitzt ResNet mit elf Schichten die komplexeste Struktur der drei Ansätze, was eine höhere Lernrate erfordert verglichen zu einer einfacheren Netzstruktur wie z. B. einem MLP mit drei verdeckten Schichten [87]. Angelehnt an Nabi wird die Hypothese aufgestellt, dass ResNet mit einer höheren Lernrate im Bereich von 0,01 bis 0,9 bessere Ergebnisse für den Recall erzielen wird [87]. Dies ist ein Grund dafür, dass die obere Intervallgrenze der Lernrate für die Zufallssuche der Parameter von ResNet auf einen vergleichsweisen hohen Wert von 0,9 eingestellt wird.

H2: Eine hohe Anzahl der Neuronen pro Schicht wird sich positiv auf den Recall vom LSTM-Ansatz auswirken

Lara-Benitez et al. konnten u. a. feststellen, dass LSTM-Netze mit hoher Anzahl von Neuronen (128) und zwei LSTM-Schichten bessere Genauigkeiten zur Vorhersage von Zeitreihendaten erzielen können [83]. Zudem begünstigt eine hohe Neuronenanzahl pro verdeckte Schicht in LSTM-Netzen die Leistungsfähigkeit, wenn der Datensatz stark abstrahiert werden kann [65]. Darauf basierend wird die Hypothese aufgestellt, dass der LSTM-Ansatz mit einer Anzahl von Neuronen pro Schicht höher als 50 einen höheren Recall-Score erzielen wird.

H3: Aufgrund der Unterangepasstheit wird sich eine komplexere Modellarchitektur positiv auf die allgemeine Leistungsfähigkeit des MLP-Ansatzes auswirken, allerdings wird diese nicht besser sein als beim ResNet- und LSTM-Ansatz

Nach erster Analyse scheint der MLP-Ansatz unter dem Problem des Underfitting zu leiden, weshalb erwartet wird, dass eine Erhöhung der Modellkomplexität einen positiven Einfluss auf die Ergebnisse haben wird. Aufgrund dessen, dass sowohl Trainings- und Validierungsergebnisse der ersten Trainingsdurchläufe bei dem MLP-Ansatz jedoch deutlich schlechter sind als bei dem ResNet- und LSTM-Ansatz, wird nicht davon ausgegangen, dass der MLP-Ansatz nach der Parametersuche bessere Ergebnisse erzielen wird als einer der übrigen Ansätze.

5.4.1 Ergebnisse der Hyperparametersuche

Als Ergebnisse der Hyperparametersuche dienen die jeweils besten hervorgebrachten Modelle. Zur Bewertung der Stabilität werden diese wie zuvor dreimal mit verschiedenen Datendurchmischungen trainiert. Mittelwert und Standardabweichung der jeweiligen Validierungsscores von Fehler, Recall und MCC sind dazu in Tabelle 5-11 aufgelistet.

Tabelle 5-11: Ergebnisse der Hyperparametersuche

Metrik	ResNet		LSTM		MLP	
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
Fehler	1,669	0,007	**0,2608**	**0,0056**	2,2137	0
Recall	0,815	0,0482	**0,9104**	0,0067	0,0909	**0**
MCC	0,8438	0,0087	**0,9117**	0,0036	0	**0**

Die Ergebnisse zeigen für ResNet und LSTM eine Verbesserung der Leistungsfähigkeit. Für ResNet beträgt die neue Anzahl der Filter pro Schicht im ersten Block 121, im zweiten Block 97 und im dritten Block 59. Die Batchgröße hat sich auf 83 und die Lernrate auf rund 0,002 erhöht. Für den LSTM-Ansatz wurde ein Modell mit 2 Schichten und 124 Neuronen pro Schicht ermittelt. Die Batchgröße beträgt 112 und die Lernrate beträgt rund 0,0036. Der Recall-Score des ResNet-Modells ist in Abbildung 5-13 als blaue Kurve dargestellt. An den Ergebnissen zeigt sich, dass eine höhere Batchgröße und Lernrate einen positiven Einfluss auf die Leistungsfähigkeit von ResNet nimmt. Allerdings bleibt die Batchgröße hinter dem erwarteten Mindestwert von 128 zurück. Darüber hinaus wurde die Erwartung, dass eine Lernrate zwischen 0,01 und 0,9 bessere Ergebnisse hervorbringen würde, nicht erfüllt. Deshalb wird H1 falsifiziert. Die Wahrheitsmatrix zeigt außerdem, dass die optimierte Konfiguration alle Manöver erkennen kann. Des Weiteren scheint eine abnehmende Anzahl der Filter pro Block

förderlich für die Leistungsfähigkeit des ResNet-Modells zu sein. Dieser Eindruck wird durch das zweitbeste Ergebnis der Parametersuche bestärkt (1. Block: 252 Filter, 2. Block: 182 Filter, 3. Block: 32 Filter; Fehler train/val: 1,7306/1,7258; Recall train/val: 68,3 % / 68,95 %). Der Recall-Score des zweitbesten Modells ist in Abbildung 5-13 als grüne Kurve dargestellt.

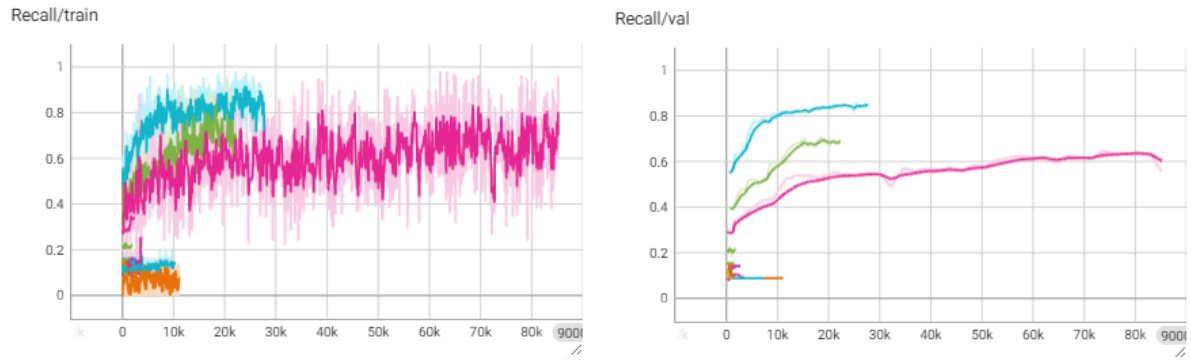


Abbildung 5-13: Recall-Scores Parametersuche ResNet

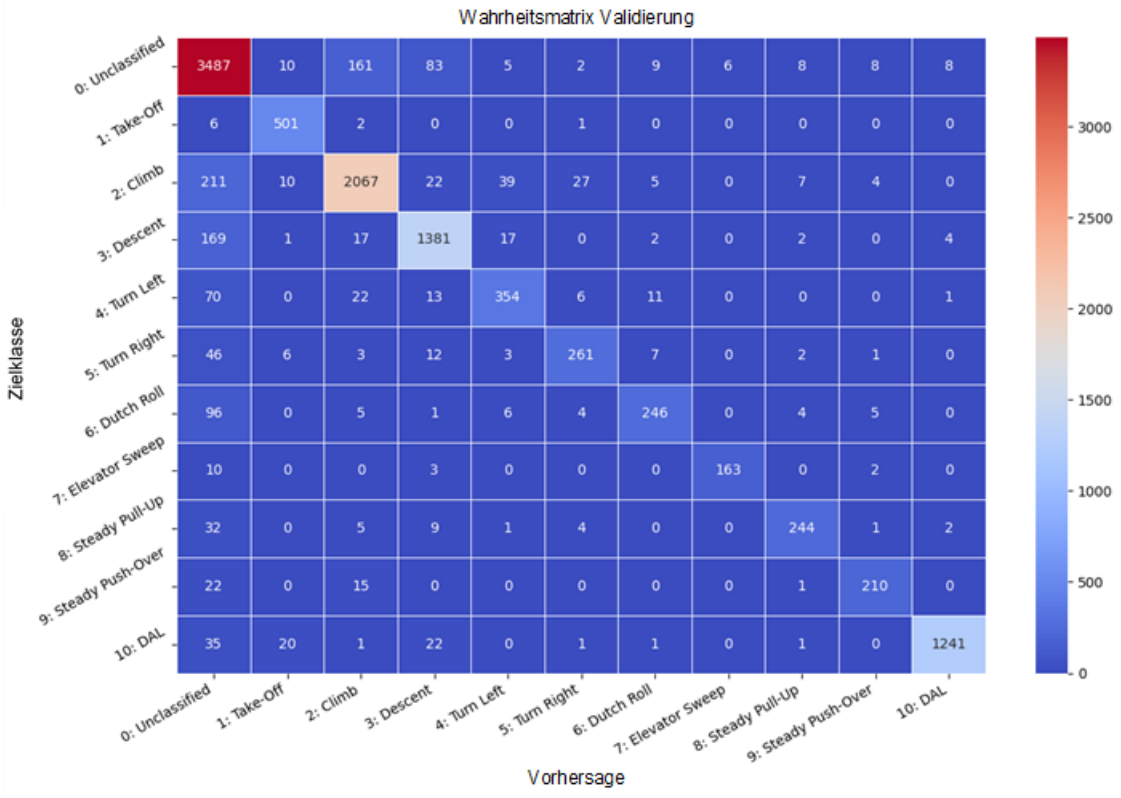


Abbildung 5-14: Wahrheitsmatrix Parametersuche des besten ResNet-Modells

Bei der Betrachtung der Ergebnisse für den LSTM-Ansatz fällt auf, dass die besten drei Modellkonfigurationen eine Lernrate zwischen 0,003 und 0,004 und eine Anzahl der Neuronen pro Schicht von mindestens 93 verwendet haben. Daher kann H2 verifiziert werden. Die Trainingsverläufe der LSTM-Parametersuche sind in Abbildung 5-15 dargestellt. Die

Wahrheitsmatrix des optimierten Modells zeigt keine Auffälligkeiten im Vergleich zu Abbildung 5-9.

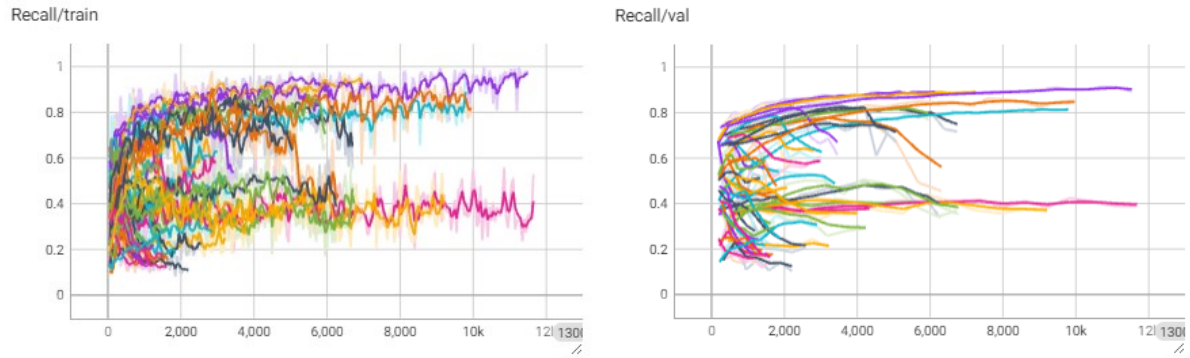


Abbildung 5-15: Recall-Scores Parametersuche LSTM

Die Ergebnisse der Parametersuche konnten für den MLP-Ansatz weder Veränderungen noch Verbesserungen der Leistungsfähigkeit hervorbringen. Daher wird H3 falsifiziert. Nach erfolgter Parametersuche werden daher die Ergebnisse folgender Konfigurationen zum Vergleich der Ansätze verwendet:

Tabelle 5-12: Modellhyperparameter Vergleich DL-Ansätze

Parameter	ResNet	LSTM	MLP
Schichten	3 Blöcke x 3 Schichten	2	3
Neuronen/Filter je Schicht	1. 3 x 121 2. 3 x 97 3. 3 x 59	124	1. 16 2. 32 3. 64
Batchgröße	83	112	16
Lernrate	0,002	0,0036	0,001
Fehlerfunktion	Cross-Entropy	Cross-Entropy	Cross-Entropy
Optimierer	Adam	Adam	Adam

5.4.2 Testergebnisse

Die aufgeführten Modelle werden abschließend auf den Testdaten evaluiert. Die Ergebnisse der Testdurchläufe sind in Tabelle 5-13 zusammengefasst. Überraschenderweise liefert der LSTM-Ansatz auf dem Testdatensatz deutlich schlechtere Ergebnisse verglichen zum Trainingsprozess. Verglichen dazu liefert er ResNet-Ansatz an dieser Stelle zwar bessere

Testergebnisse, allerdings ist dabei ebenfalls eine hohe Differenz zu den Trainingsergebnissen festzustellen. Aus diesem Grund kann angenommen werden, dass der LSTM- und ResNet-Ansatz auf die Trainingsdaten überangepasst sind. Die Testergebnisse für den MLP-Ansatz zeigen nach wie vor keine Verbesserung, was nach den zuvor beschriebenen Trainingsergebnissen allerdings erwartbar war. Auffällig ist jedoch, dass zum Test-Recall die Differenz des MCC zwischen ResNet und LSTM vergleichsweise gering ausfällt.

Tabelle 5-13: Testergebnisse für ResNet, LSTM und MLP

Metrik	ResNet		LSTM		MLP	
	$\bar{x}$	σ	$\bar{x}$	σ	$\bar{x}$	σ
Fehler	**1,978**	0,036	3,351	0,211	2,152	**0**
Recall	**0,6015**	0,0262	0,4768	0,0234	0,0909	**0**
MCC	**0,445**	0,042	0,409	0,01	0	**0**

Da in den ResNet- und LSTM-Modellen bereits Maßnahmen zur Verringerung von Overfitting angewandt werden (z. B. Dropout, Batch Normalisierung, Pooling) wird vermutet, dass die hohe Varianz möglicherweise mit Fehlern im Bereich der Datenvorverarbeitung und speziell mit der Auswahl der Parameter begründet werden kann. Die 19 ausgewählten Messparameter für das Training der Modelle könnten z. B. irrelevante Parameter enthalten, die zum Overfitting der Modelle beitragen. Die Sequenzierung der Daten wurde zudem mit einer kleinen Fenstergröße (32 Datenpunkte = 3,2 Sek.) und einer mittleren Überlappung der Datensequenzen (50 %) durchgeführt. Daher ist es wahrscheinlich, dass manche für die Klassifizierung wichtigen manövertypischen Zusammenhänge in den Daten möglicherweise nicht repräsentiert sind, da sie über größere Zeiträume auftreten, die länger als 3,2 Sekunden dauern. Zudem kann die halbe Überlappung der Datenfenster dazu führen, dass das Modell zu wenige verschiedene Ausschnitte der Manöver kennenlernt, sodass unbekannte Manöverausschnitte in den Testdaten möglicherweise schlechter erkannt werden. Die vergleichsweise geringe Differenz der MCC-Scores deutet darauf hin, dass der schlechtere Recall-Score des LSTM-Ansatzes dadurch zustande kommt, dass das Modell auf den unterrepräsentierten Manövern im Datensatz schlechter funktioniert. Dabei fallen Falschklassifizierungen stärker ins Gewicht und ziehen den Recall-Score folglich schneller runter. Ein Blick in die Wahrheitsmatrizen der beiden Ansätze bestätigt diese These (Abbildung 5-16, Abbildung 5-17). Dabei ist zu erkennen, dass dies vor allem bei den Manövern Turn

Right, Turn Left, Elevator Sweep und Steady Pull-Up der Fall ist. Aus dem Muster fallen dagegen die Manöver Dutch Roll und Steady Push Over. Während das Steady Push-Over Manöver von beiden Ansätzen besser erkannt wird, klassifizieren beide Ansätze das Dutch Roll Manöver häufiger als Unclassified. Grund dafür ist möglicherweise die erhöhte Variabilität des Dutch Roll Manövers in den Daten.

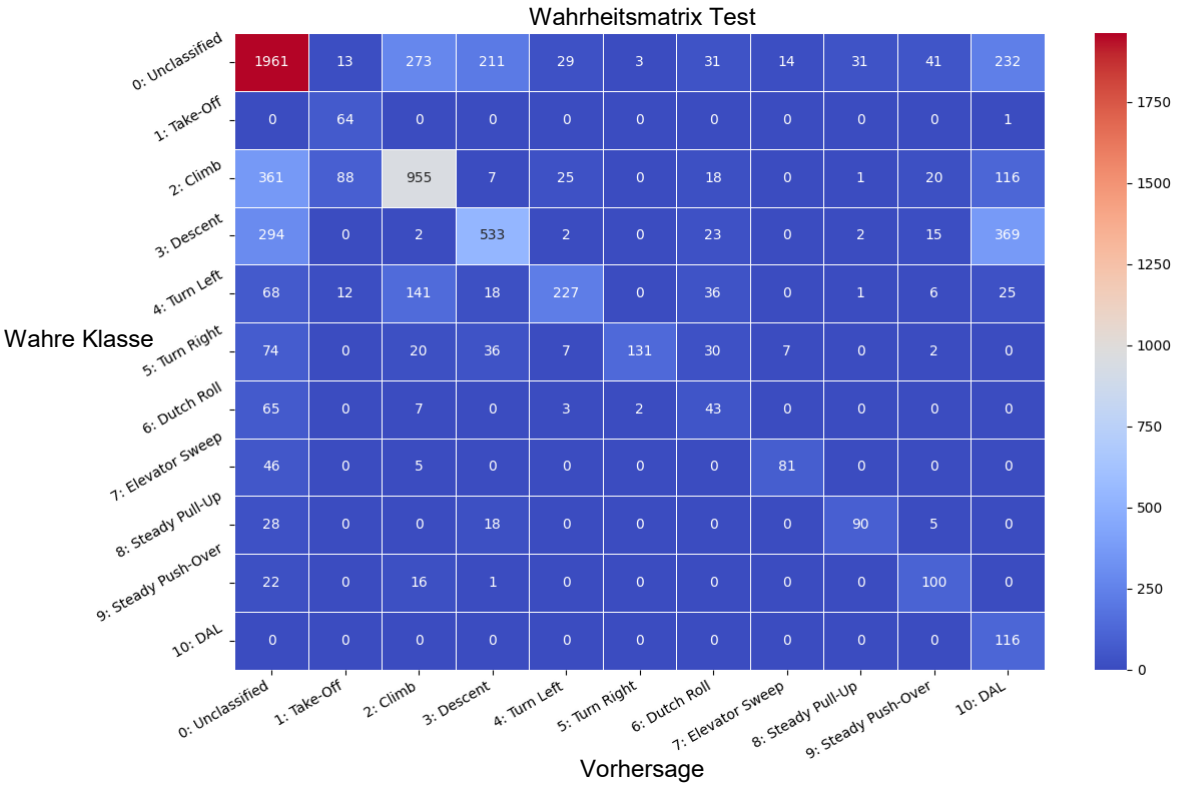


Abbildung 5-16: Wahrheitsmatrix Test ResNet

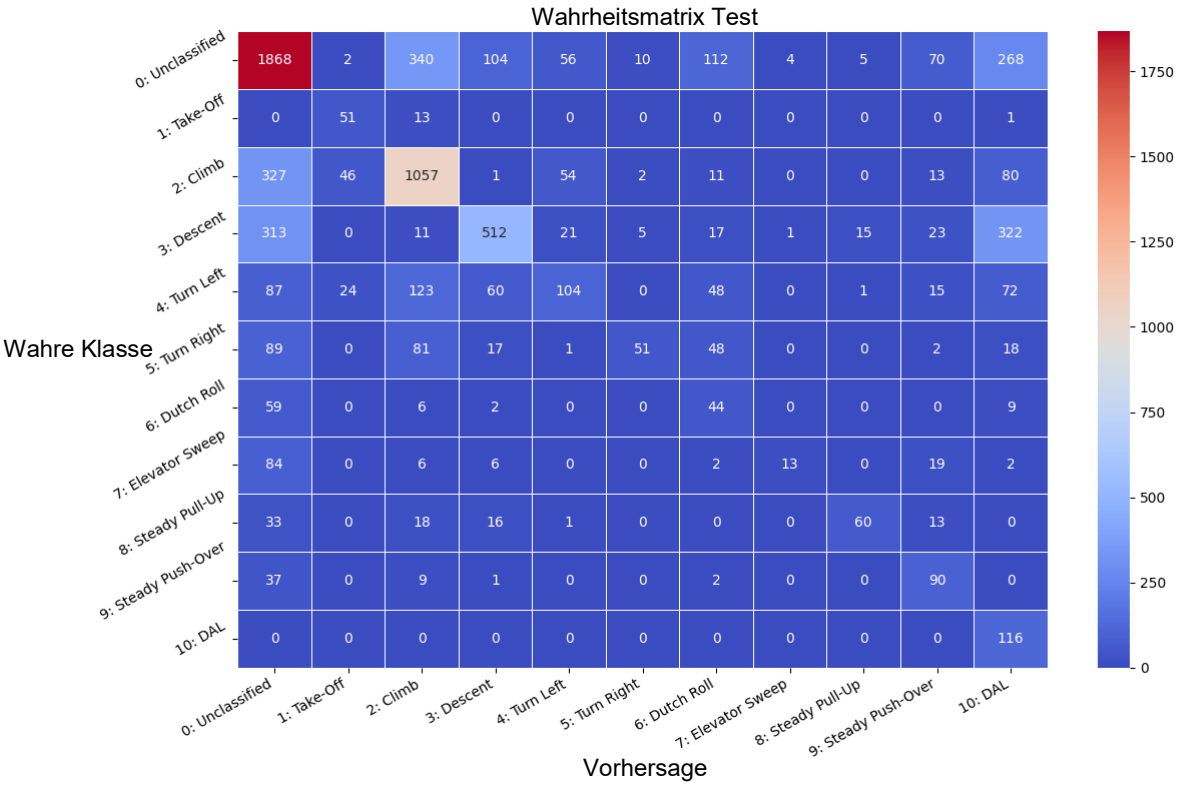


Abbildung 5-17: Wahrheitsmatrix Test LSTM

Die insgesamt schlechten Ergebnisse des MLP-Ansatzes lassen zum einen darauf schließen, dass die manuell entwickelten physikalischen Parameter den Datenraum möglicherweise zu stark reduziert haben. Des Weiteren reicht die Komplexität der binären Merkmalsvektoren wahrscheinlich nicht aus, um die Charakteristika der verschiedenen Manöver eindeutig zu identifizieren und voneinander zu unterscheiden.

5.5 Bewertung der Ansätze

Um die verschiedenen DL-Ansätze zu bewerten, werden die Bewertungsmaßstäbe der unterschiedlichen Ansätze berechnet. Darauf aufbauend wird zum Abschluss des Vergleichs für jedes Kriterium jeweils eine Punktzahl zwischen 1 und 10 vergeben. Die Punktzahl wird anschließend je nach Kriterium gewichtet und zu einem Gesamtscore für jeden Ansatz zusammengerechnet. Anhand dessen wird der beste Ansatz ausgewählt, um diesen weiter zu optimieren. Die Ermittlung der in Kap. 5.1.2 aufgeführten Bewertungsmaßstäbe wird in der folgenden Tabelle aufgelistet. Die besten Werte sind dabei fett markiert.

Tabelle 5-14: Bewertungsmaßstäbe für ResNet, LSTM und MLP

Kriterium	Bewertungsmaßstab	ResNet	LSTM	MLP
Korrektheit im Entwicklungsprozess	$Recall_{val}$	0,815	**0,9104**	0,0909
Korrektheit im Einsatz	$Recall_{test}$	**0,6015**	0,4768	0,0909
Relevanz	$Recall_{val} - Recall_{test}$	0,2135	0,4336	**0**
Robustheit	$Dutch\ Roll\ Recall_{val}$	0,5250	**0,7149**	0
Stabilität	$Recall_{val\_\sigma}$	0,0482	0,0067	**0**
Fairness	$Class\ Recall_{val\_max} - Class\ Recall_{val\_min}$	0,4085	**0,2542**	0,9999
Interpretierbarkeit	$n_{layer} * n_{neur} + n_{wght}$	401.786	197.519	**3.675**

Tabelle 5-15: Punktevergabe Vergleich ResNet, LSTM, MLP

		ResNet		LSTM		MLP	
	Gewichtung	Bewertung	Wert	Bewertung	Wert	Bewertung	Wert
Korrektheit im Entwicklungsprozess	10.71%	8	0.86	9	0.96	1	0.11
Korrektheit im Einsatz	19.64%	6	1.18	5	0.98	1	0.20
Relevanz	17.86%	8	1.43	6	1.07	10	1.79
Robustheit	19.64%	5	0.98	7	1.38	0	-
Stabilität	21.43%	9	1.93	10	2.14	10	2.14
Fairness	7.14%	6	0.43	7	0.50	0	-
Interpretierbarkeit	3.57%	1	0.04	5	0.18	10	0.36
	Summe		**6.84**		**7.21**		**4.59**

Tabelle 5-15 zeigt die abschließende Punktevergabe im Rahmen des Vergleichs der Ansätze. Aus dem Vergleich geht der LSTM-Ansatz knapp vor ResNet als beste Lösung hervor. Der MLP stellt sich dagegen klar als ungeeignetste Lösung heraus. Dementsprechend wird das LSTM-Modell im weiteren Verlauf durch die Anwendung verschiedener Verbesserungsmaßnahmen für die automatisierte Manöverdetektion weiterentwickelt. ResNet und MLP werden im weiteren Verlauf nicht mehr berücksichtigt.

6 Experimentierphase LSTM

Nachdem das gewählte LSTM-Modell aus dem Vergleich als am geeignetsten hervorging, werden in diesem Abschnitt verschiedene Experimente zur weiteren Optimierung des Modells behandelt. Zum Abschluss erfolgt eine erneute Evaluation des weiterentwickelten LSTM-Modells auf dem Testdatensatz.

Als Hauptproblem des entwickelten LSTM-Modells wurde identifiziert, dass es eine hohe Varianz aufweist. Das heißt es liefert auf den noch unbekannten Testdaten deutlich schlechtere Ergebnisse als auf den Validierungsdaten aus dem Trainingsprozess was ein starkes Indiz für ein „Overfitting“ des Modells ist. Als mögliche Ursache wurde dafür die Dimensionalität der Daten und dabei speziell die Auswahl der Parameter genannt. Als wichtiges Hilfsmittel ist in diesem Zusammenhang die Ermittlung der sog. „Permutation Feature Importance“ (kurz: „Feature Importance“, dt.: Permutation der Parameterrelevanz oder Parameterrelevanz) zu nennen. Dabei werden die Werte der verschiedenen Parameter in einem Datensatz nacheinander durchmischt und der jeweilige Fehler des Modells berechnet. Steigt der Fehler an, ist dies ein Indiz dafür, dass der durchmischte Parameter eine hohe Relevanz für die Vorhersage des Modells besitzt. [88]

Die daraus resultierenden Ergebnisse zur Relevanz der Parameter sind in Abbildung 6-1 dargestellt. Anhand dessen ist zu erkennen, dass neun der 19 Parameter einen Wert für die Feature Importance von über 0,1 erreichen und sich damit von den übrigen Parametern abheben. Dazu zählen die Parameter sys_time, CPT_L, IRS_ALT, IRS_HDG, IRS_LON, IRS_THE, IRS_VV, TAS und TAT. Daher wird das LSTM-Modell im Anschluss nur mit diesen Parametern aus dem Datensatz trainiert. Die Trainingsresultate werden im Anschluss mit denen des Referenzmodells aus Kap. 5.4 verglichen.

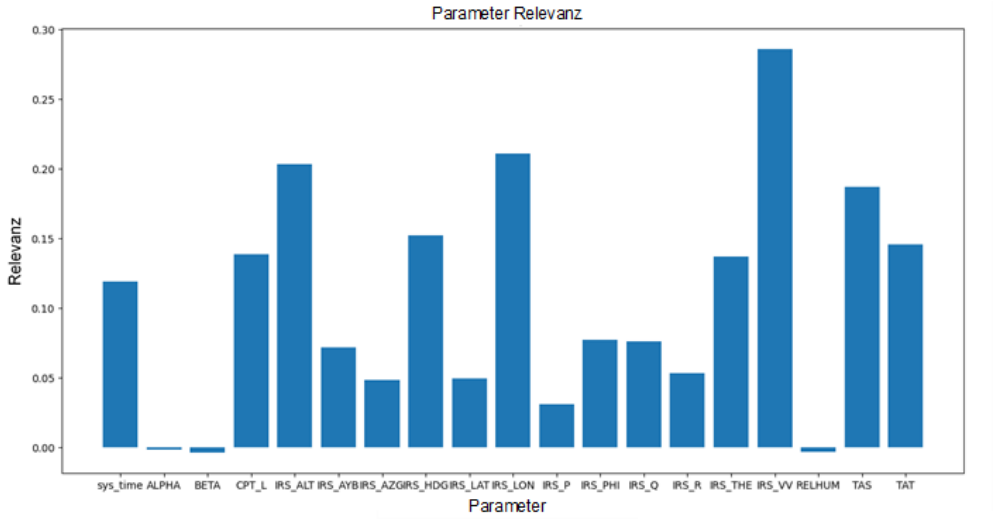


Abbildung 6-1: Parameter Relevanz für 19 Trainingsparameter

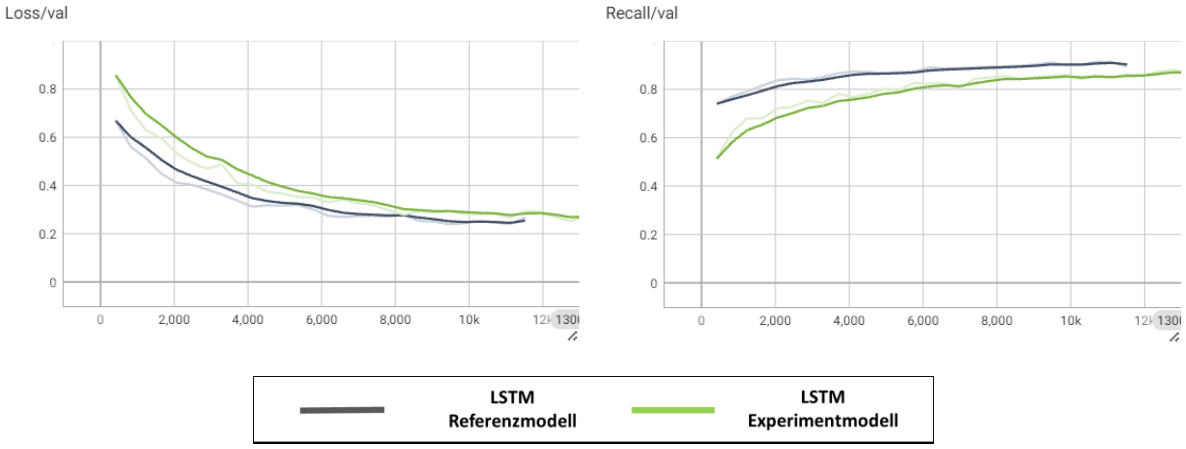


Abbildung 6-2: Fehler- und Recallverlauf von LSTM-Referenzmodell und LSTM-Modell mit reduziertem Parameterset

Anhand der Ergebnisse kann festgestellt werden, dass sich die Ergebnisse im Vergleich zum Referenzmodell leicht verschlechtern (Fehler +0,0172; Recall -3,41 %). Die kleine Veränderung lässt darauf schließen, dass die entfernten Parameter tatsächlich keinen starken Einfluss auf die Vorhersage des Modells hatten und ist erwartbar, da die entfernten Parameter trotzdem eine geringe Auswirkung auf die Ergebnisse des Modells haben.

Im Falle einer hohen Varianz hat sich außerdem laut Ng die Vergrößerung des Trainingsdatensatz als weitere Optimierungsmaßnahme bewährt [48]. Um zunächst eine erneute Datenakquise und einen aufwändigen Labelingprozess zu vermeiden, wird dafür alternativ die Überlappung des Sliding-Window vergrößert, um zusätzliche Datenbeispiele zu generieren. So wird in einem weiteren Experiment der Einfluss der Überlappung der Datensequenzen untersucht, indem diese auf 80 % erhöht wird. Durch die Herangehensweise den Datensatz in Subsequenzen aufzuteilen, kann das LSTM-Netz immer nur die

Zusammenhänge innerhalb einer Datensequenz zur Klassifizierung nutzen. Die schlechten Testergebnisse können daher ebenfalls auf einer schlecht gewählten Fenstergröße des Sliding-Window beruhen. Aus diesem Grund wird in einem weiteren Experiment der Einfluss der Fenstergröße des Sliding-Window untersucht, indem diese auf 64 Datenpunkte, also 6,4 Sekunden, verlängert wird. Der Wert für die Fenstergröße orientiert sich an der Länge des kürzesten gelabelten Manövers, die 7,3 Sekunden beträgt. Damit soll möglichst vermieden werden, dass die Anzahl der Datensequenzen, in denen mehrere gelabelte Manöver enthalten sind, ansteigt und dadurch falsche Muster von dem LSTM-Modell für bestimmte Klassen gelernt werden.

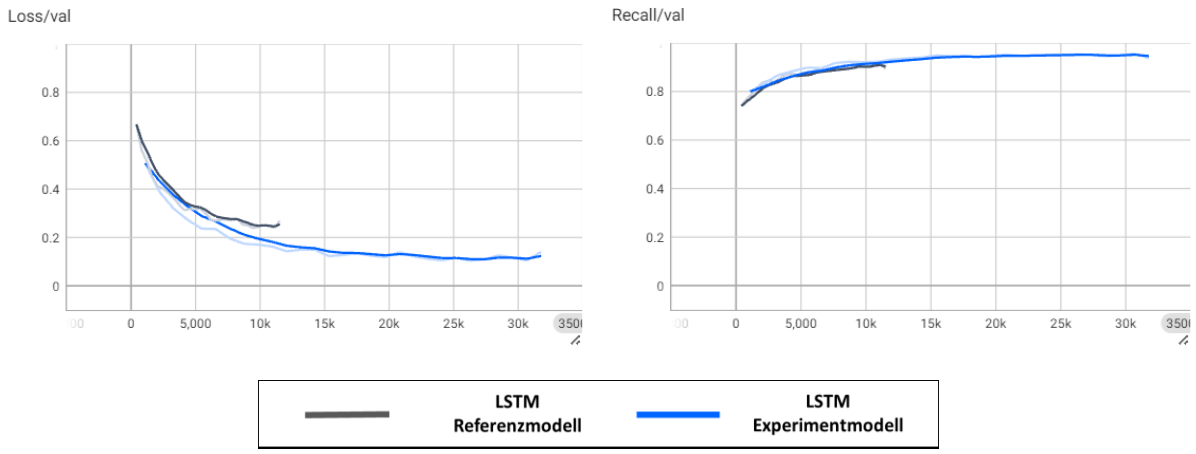


Abbildung 6-3: Fehler- und Recallverlauf LSTM-Referenzmodell und LSTM-Modell mit Überlappung 80 %

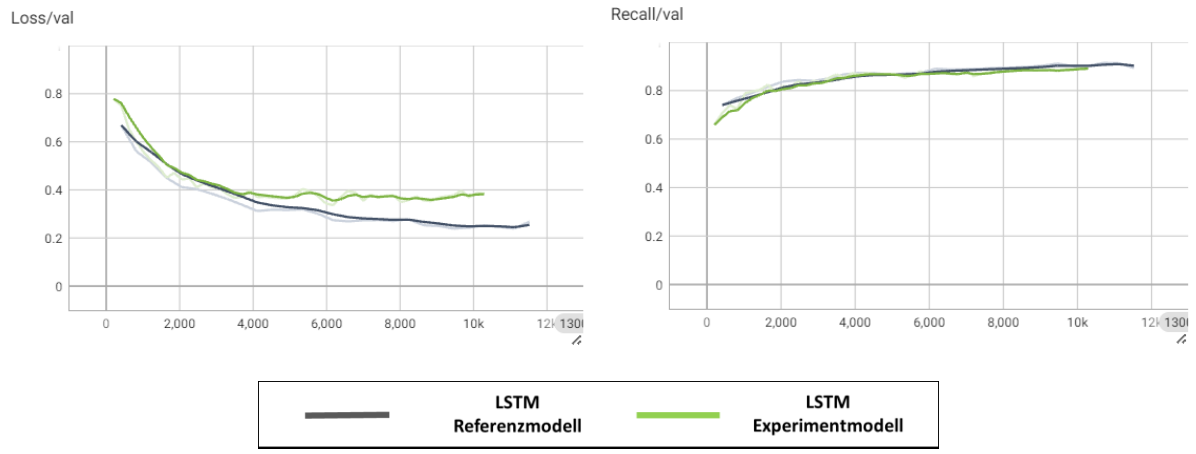


Abbildung 6-4: Fehler- und Recallverlauf LSTM-Referenzmodell und LSTM-Modell mit Fenstergröße 64

Die Ergebnisse der beiden beschriebenen Experimente zeigen, dass die erhöhte Überlappung einen deutlichen positiven Effekt auf die Leistungsfähigkeit des Modells bei der Validierung nimmt (Fehler -0,131; Recall +4,31 %). Aus diesem Grund kann angenommen werden, dass

die Maßnahme zur Erweiterung des Datensatzes erfolgreich ist. Darauf aufbauend kann erwartet werden, dass sich dies auch positiv in den Testergebnissen widerspiegeln wird. Die Erhöhung der Fenstergröße konnte dagegen keine Verbesserung der Validierungsergebnisse erzielen (Fehler +0,128; Recall -1,24 %) und gelangt früher bei ca. 4000 Trainingsschritten in den Bereich des Übertrainings. Dadurch, dass das Modell mit erhöhter Fenstergröße weniger Trainingsbeispiele lernt, ist dieser Effekt jedoch erwartbar.

Als viertes Experiment wird der Einfluss der Strategie zur Auswahl des Labels einer Sequenz untersucht. Bisher wurde immer das Label des letzten Messpunktes einer Datensequenz als Label für die gesamte Sequenz verwendet. Diese Strategie führt jedoch speziell in den Randbereichen der Manöver dazu, dass ein Label für eine Sequenz ausgewählt wird, dessen Messpunkte überwiegend mit einem anderen Label markiert wurden. Daher wird in diesem Experiment der Einfluss der Labelauswahl pro Sequenz geprüft. Dafür wird das Label einer Sequenz anhand des meist verwendeten Labels einer Datensequenz ausgewählt [89].

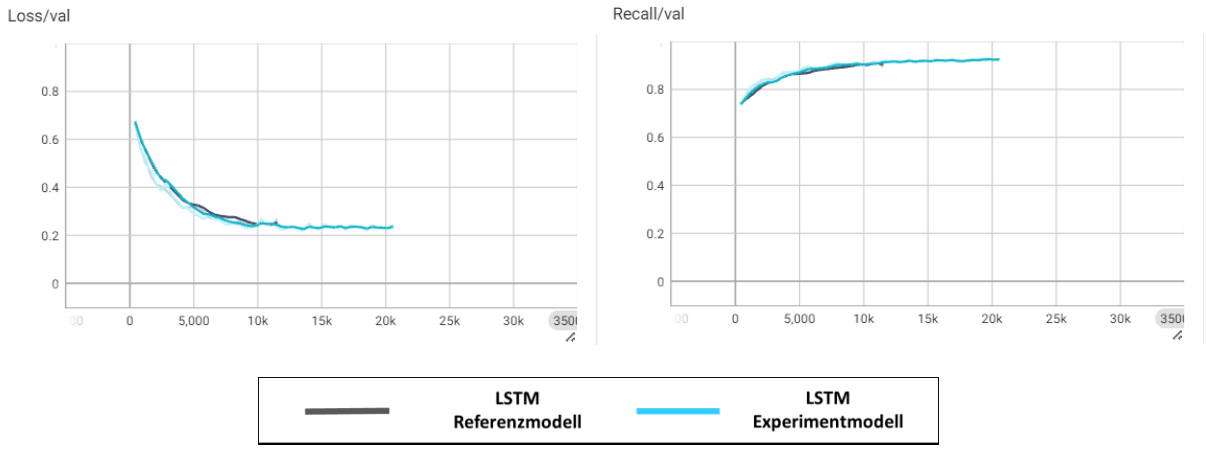


Abbildung 6-5: Fehler- und Recallverlauf LSTM-Referenzmodell und LSTM-Modell mit angepasster Sequenzlabelauswahl

Aus dem Experiment kann festgestellt werden, dass der Recall gegenüber dem LSTM-Referenzmodell leicht angestiegen ist (+2,2 %) Der Fehler ist parallel dazu auf 0,2362 (-0,0182) gesunken. Daher bestätigt sich der zu erwartende positive Effekt zunächst leicht.

Abschließend werden alle experimentierten Modelle erneut auf den Testdaten evaluiert. Die Ergebnisse sind in Tabelle 6-1 aufgelistet. Daraus wird deutlich, dass die Verbesserungsmaßnahme „Sequenzlänge 64“ (+ 10,55 %) die größte Verbesserung des Recalls aus den Testergebnissen erzielen konnte. Zudem konnten die Maßnahme „Überlappung 80 %“ (+ 2,04 %) und „Labeling Strategie“ (+ 0,8 %) ebenfalls eine leichte Verbesserung des Recall erzielen. Auffällig ist, dass die Maßnahme „Parameter Relevanz“ die Leistungsfähigkeit auf den Testdaten gemessen am Fehler (+ 0,4886) und MCC (- 0,1405) stark verschlechtert hat. Allerdings ist der Recall nahezu gleichgeblieben.

Tabelle 6-1: Testergebnisse LSTM-Optimierung

Experiment	Fehler	Recall	MCC
Referenz	3,3806	0,4926	0,4184
Parameter Relevanz	3,8692	0,4891	0,2779
Überlappung 80 %	2,8401	0,513	0,4263
Sequenzlänge 64	**2,7758**	**0,5981**	**0,4429**
Labeling Strategie	3,1034	0,5006	0,3665

Die Verschlechterung durch das Experiment „Parameter Relevanz“, deutet darauf hin, dass das Modell Muster in Parametern erkennt, die sich schlecht verallgemeinern lassen. So sind die Flugversuche der „DLR Summer School“ teils ähnlich aufgebaut, sodass nicht nur die Flugmanöver meist in annähernd gleicher Reihenfolge geflogen werden, sondern ebenso die Flüge vereinzelt zu ähnlichen Tageszeiten oder auf nahe gelegenen Flugrouten stattfinden. Das steigert das Risiko, dass manche Flugmanöver wahrscheinlich zu ähnlichen Tageszeiten oder bei ähnlichen Breiten- und Längengraden. Dadurch würden die Modelle diese zeitlichen und örtlichen Muster ebenfalls lernen, was wiederum einen Anteil des starken Übertrainings erklären würde. Daher ist es plausibel, dass dazu z. B. die Parameter „sys_time“, „IRS_LON“ und „IRS_LAT“ maßgeblich beitragen. Daneben zeigt die Verbesserung durch die Anpassung der Sequenzlänge, dass weitere Experimente zu dessen Optimierung womöglich weitere Erfolge erzielen können. Allerdings ist dabei zu beobachten, ab wann die Zeitgrenzen der Manöver durch die Sequenzlänge so weit überschritten werden, sodass den Manövern ebenfalls zu viele irrelevante Datenpunkte zugeordnet werden.

7 Kritische Betrachtung

Bei Betrachtung der Ergebnisse des Modellvergleichs konnte festgestellt werden, dass das MLP-Modell nicht nur eine deutlich schlechtere Leistungsfähigkeit aufweist als das ResNet- und LSTM-Modell, sondern dazu keines der zu identifizierenden Manöver erkennt. Zwar kann dies ein Indiz dafür sein, dass sich die Modellarchitektur des MLP für die vorliegende Problemstellung generell nicht eignet, allerdings können hierfür auch Fehler in der Vorverarbeitung der Flugdaten verantwortlich sein. So ist es plausibel, dass es den Merkmalen der binären Merkmalsvektoren für das MLP an ausreichender Komplexität fehlt, um die Flugmanöver in den Messdaten eindeutig identifizieren zu können. Weil dadurch die Voraussetzungen für die verschiedenen Modellansätze möglicherweise stärker voneinander abweichen, verzerrt dies die Vergleichbarkeit der Modelle. Positiv hervorzuheben ist, dass die verschiedenen Ansätze nicht allein auf Grundlage eines Scores zur Beurteilung der Leistungsfähigkeit verglichen wurden. Stattdessen wurde ein aktuelles Qualitätsmodell aus der Forschung nach Siebert et al. verwendet, um neben der Leistungsfähigkeit unter anderem die Stabilität oder Voreingenommenheit in der Beurteilung eines Modells zu beachten. Zudem wurde das Qualitätsmodell mit Methoden einer Nutzwertanalyse verbunden, wodurch die Ergebnisse weiter an Aussagekraft gewinnen.

8 Zusammenfassung und Ausblick

Nach anfänglicher Recherche zu bestehenden Lösungsansätzen aus den Bereichen der Flugdatenverarbeitung und Zeitserienklassifizierung im Allgemeinen, wurden basierend auf den Erkenntnissen ein ResNet-, ein LSTM und ein MLP-Ansatz als Vergleichsmodelle ausgewählt. Zudem wurde ein Datensatz von realen Flugversuchsdaten des DLR zusammengestellt, der für das Training und die Evaluation der gewählten Ansätze verwendet werden sollte. Da die Flugmanöver in den ausgewählten Daten noch nicht markiert waren, wurden im Rahmen eines manuellen Labelingprozesses sechs Flugzustände und vier komplexe Flugmanöver in den Daten mit Labels gekennzeichnet. Aus einer detaillierten Datenanalyse wurden anschließend die Rahmenbedingungen für die Datenvorverarbeitung, welche im Anschluss durchgeführt wurde, abgeleitet. Auf Grundlage der vorverarbeiteten Daten wurden anschließend die ausgewählten DL-Modelle zuerst mit den Trainingsdaten trainiert. Anschließend wurden dazu im Rahmen einer Parametersuche Experimente mit unterschiedlichen Modell- und Trainingskonfigurationen durchgeführt. Dafür wurde die Methode der Zufallssuche angewandt. Die Zufallssuche war für das MLP-Modell nicht erfolgreich, konnte allerdings für das ResNet- und LSTM-Modell bessere Konfigurationen hervorbringen. Für den Fall des ResNet-Modells hat sich dabei eine abnehmende Anzahl von Merkmalsfiltern pro Residual Block in Kombination mit einer hohen Batchgröße als wirksame Einstellung herausgestellt. Das verbesserte LSTM-Modell hat sich außerdem durch eine erhöhte Anzahl von versteckten Einheiten pro Schicht und eine erhöhte Batchgröße ausgezeichnet. Anschließend wurden die neuen Modelle gemeinsam mit der ursprünglichen MLP-Konfiguration anhand eines Testdatensatzes evaluiert. Um eine Antwort auf die Forschungsfrage zu finden, wurden die verschiedenen DL-Ansätze anschließend unter Verwendung von Qualitätskriterien für ML-Modelle bewertet. Aus der Bewertung der Modelle ging hervor, dass unter den ausgewählten DL-Ansätzen der LSTM- und der ResNet-Ansatz sich am ehesten als Lösung einer automatisierten Flugmanöverdetektion auf Grundlage von Flugmessdaten eignen. Allerdings zeigten die Testergebnisse, dass zur Verwendung der Modelle im Einsatz eine weitere Optimierung notwendig ist. Um dafür mögliche Optimierungsansätze zu identifizieren, wurde der LSTM-Ansatz für eine weitere Experimentierphase ausgewählt. Dabei wurde anhand von vier Trainingsexperimenten versucht festzustellen, welche Verbesserungsmaßnahmen sich für eine weitere Optimierung des LSTM-Ansatzes als aussichtsreich darstellen. Dabei wurde sich primär auf die Vorverarbeitung des Datensatzes spezialisiert. Als ein vielversprechender Verbesserungsansatz stellte sich dadurch die Variation der Datensequenzlänge heraus. Um ein Modell in den Einsatz zu bringen, haben auch die Testergebnisse der Experimentierphase

bisher allerdings noch nicht ausgereicht. Außerdem wurden die Testläufe bisher nur auf Testdaten eines Forschungsflugzeuges durchgeführt, weswegen noch unbekannt ist, ob sich der Ansatz auf Flugdaten anderer Flächenflugzeuge übertragen lässt. Letztendlich kann festgestellt werden, dass sich die untersuchten Ansätze in der gewählten Form noch nicht für den Realeinsatz als automatisierte Flugmanövererkennung eignen. Allerdings geben die Trainingsergebnisse einen Hinweis auf das Potenzial des ResNet- und LSTM-Ansatzes. Daher ist durch eine weitere Erforschung der beiden Lösungsvorschläge der mögliche Einsatz im Echtzeitbetrieb als realistisch einzuschätzen.

Die vorliegende Arbeit trägt zum einen zur Erforschung der automatisierten sensorbasierten Erkennung von Flugmanövern mithilfe von tiefen Neuronalen Netzen bei. Dabei liegt der Fokus auf der Verwendung der DL-Algorithmen als Mehrklassen-Klassifikatoren, um verschiedene komplexe Flugmanöver zu identifizieren, wovon manche der Längsbewegung und andere der Seitenbewegung eines Flugzeugs zugeordnet werden. Die vergleichende Betrachtung beinhaltet zudem ein ResNet-Modell für die Flugmanöverdetektion, das in dem Bereich der sensorbasierten Klassifizierung von Flugmanövern kaum erforscht ist. Zum anderen knüpft die vorliegende Arbeit an die Forschung von Siebert et al. zu Bewertungsmethoden für ML-Modelle an und verankert die modellspezifischen Qualitätskriterien in Elementen einer Nutzwertanalyse und wendet diese auf tiefe Neuronale Netze an. Neben vereinzelten Bewertungsmaßstäben nach Siebert et al. werden dabei außerdem andere vereinfachte Metriken zur Bewertung der Qualitätskriterien verwendet. Darüber hinaus stellt die vorliegende Arbeit eine Basis zur Entwicklung eines funktionalen Ansatzes zur Erkennung von Flugmanövern in den Messsignalen der DLR-Forschungsflotte bereit. Damit wird ein Beitrag zur Entwicklung des digitalen Zwillings eines Forschungsflugzeuges im Rahmen des DigECAT-Projekts geleistet.

Zum Ende der Experimentierphase wurde deutlich, dass die Reduktion der ausgewählten Datenparameter eher eine Verschlechterung statt einer erwarteten Verbesserung des LSTM-Ansatzes hervorgebracht hat. Dies deutet auf Fehler bei der Auswahl der Datenparameter hin, weshalb dieser Aspekt in der Weiterentwicklung des Projektes nochmals untersucht werden sollte. Dabei sollte speziell der Einfluss der Parameter „sys_time“, „IRS_LAT“ und „IRS_LON“ beurteilt werden. Auch mit der Vergrößerung des Datensatzes durch eine erhöhte Überlappung des Sliding-Window konnte nach wie vor eine hohe Varianz des gewählten LSTM-Ansatzes festgestellt werden. Aufgrund dessen sollte eine Erweiterung des Datensatzes mit weiteren Flugdatensätzen für die kommende Entwicklung der automatisierten Manöverdetektion in Betracht gezogen werden. Dabei ist die Nutzung von weiteren Forschungsflugzeugen als Datenquelle denkbar, um die Variabilität der Daten zu erhöhen und

auf die Anwendung des Ansatzes für die gesamte DLR-Forschungsflotte hinzuarbeiten. Die Anpassung der Fenstergröße ging dagegen aus der Experimentierphase als wirkungsvollste Verbesserungsmaßnahme hervor. Daher ist es zukünftig sinnvoll dahingehend weitere Experimente durchzuführen, um die optimale Fenstergröße für den Anwendungsfall zu finden. Falls die genannten Maßnahmen nicht die erforderlichen Verbesserungen zeigen sollten, wäre der Einsatz von verschiedenen Ansätzen für verschiedene Gruppen von Manövern vorstellbar. Als alternativer Ansatz könnte außerdem der ResNet-Ansatz, ein hybrider Ansatz bestehend aus 1D-CNN und LSTM-Netz oder eine Betrachtung der Manöverdetektion als Problem der Bildklassifizierung mit 2D-CNNs infrage kommen. Dies zeigt, dass zahlreiche Möglichkeiten zur Optimierung des Ansatzes im Rahmen der vorliegenden Arbeit noch nicht ausgeschöpft werden konnten. Allerdings dient sie als Grundlage und erprobte Orientierungshilfe für die Weiterentwicklung und fortlaufende qualitative Bewertung der Deep Learning-Ansätze für die automatisierte Detektion von Flugmanövern.

Quellenverzeichnis

[1] F. N. Stoliker, „Introduction to flight test engineering (Introduction aux techniques des essais en vol),“ *NATO RESEARCH AND TECHNOLOGY ORGANIZATION NEUILLY-SUR-SEINE (FRANCE),* 2005.

[2] W. Borgmann, „Mission Luftfahrtforschung: Die Forschungsflugzeuge des DLR,“ JAHR MEDIA GmbH & Co. KG, 16 03 2024. [Online]. Available: https://www.aerointernational.de/industrie-technik-nachrichten/mission-luftfahrtforschung-die-forschungsflugzeuge-des-dlr.html. [Zugriff am 03 07 2024].

[3] K. Lenz, „400 Flugmanöver mit dem ISTAR bringen den Digitalen Zwilling nach vorne,“ Deutsches Zentrum für Luft- und Raumfahrt e.V., 17 05 2023. [Online]. Available: https://www.dlr.de/de/aktuelles/nachrichten/2023/02/400-flugmanoever-mit-dem-istar-bringen-den-digitalen-zwilling-nach-vorne. [Zugriff am 06 17 2024].

[4] R. Baierl und M. Stiebitz, „Potenziale des Digitalen Zwillings im Produktlebenszyklus,“ in *Smart Services: Band 1: Konzepte–Methoden–Prozesse*, Springer, 2022, p. 291–307.

[5] Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V., „Digitaler Zwilling,“ Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V., o. J.. [Online]. Available: https://www.iese.fraunhofer.de/de/leistungen/digitaler-zwilling.html. [Zugriff am 06 17 2024].

[6] C. McMahon, „Digitaler Zwilling - Anwendungsfälle,“ 18 07 2022. [Online]. Available: https://www.ptc.com/de/blogs/corporate/best-practical-digital-twin-use-cases. [Zugriff am 17 06 2024].

[7] M. Mierke, „Was sind Metadaten?,“ Heise Medien GmbH & Co. KG, 29 03 2021. [Online]. Available: https://www.heise.de/tipps-tricks/Was-sind-Metadaten-6001016.html. [Zugriff am 17 06 2024].

[8] E. Arts, M. Bäßler, S. Haufe, A. Kamtsiuris, H. Meyer, C. Pätzold, R. Schültzky und M. Tchorzewski, „Digital Twin for Research Aircraft,“ *Deutscher Luft-und Raumfahrtkongress 2022,* 2023.

[9] S. Haufe, M. Bäßler, C. Pätzold, M. Tchorzewski, H. Meyer, E. Arts und A. Kamtsiuris, „Digital Twins Storage and Application Service Hub (Twinstash),“ 2023.

[10] B. Neyshabur, S. Bhojanapalli, D. McAllester und N. Srebro, „Exploring generalization in deep learning,“ *Advances in neural information processing systems,* Bd. 30, 2017.

[11] Y. LeCun, Y. Bengio und G. Hinton, „Deep learning,“ *nature,* Bd. 521, p. 436–444, 2015.

[12] J. Lüken, „Die Geschichte des Deep Learning (2011-heute),“ LinkedIn Corporation, 09 01 2020. [Online]. Available: https://www.linkedin.com/pulse/die-geschichte-des-deep-learning-2011-heute-johannes-l%C3%BCken/. [Zugriff am 05 07 2024].

[13] L. Quintanilla, G. Warren und H. Yoshioka, „Was ist Deep Learning?,“ Microsoft Corporation, 19 04 2024. [Online]. Available: https://learn.microsoft.com/de-de/dotnet/machine-learning/deep-learning-overview. [Zugriff am 05 07 2024].

[14] J. Lu, L. Pan, J. Deng, H. Chai, Z. Ren und Y. Shi, „Deep learning for flight maneuver recognition: A survey,“ *Electronic Research Archive,* Bd. 31, p. 75–102, 2023.

[15] J. C. B. Gamboa, „Deep Learning for Time-Series Analysis,“ 07 01 2017. [Online]. Available: http://arxiv.org/abs/1701.01887. [Zugriff am 14 05 2024].

[16] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar und P.-A. Muller, „Deep learning for time series classification: a review,“ *Data mining and knowledge discovery,* Bd. 33, p. 917–963, 2019.

[17] A. Mammeri, Y. Zhao, A. Boukerche, A. J. Siddiqui und B. Pekilis, „Design of a semi-supervised learning strategy based on convolutional neural network for vehicle maneuver classification,“ in *2019 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, 2019.

[18] L. Ma und S. Tian, „A hybrid CNN-LSTM model for aircraft 4D trajectory prediction,“ *IEEE access,* Bd. 8, p. 134668–134680, 2020.

[19] V. Carlsson, *Detecting flight patterns using deep learning,* Uppsala: Uppsala universitet, 2023.

[20] I. Kontopoulos, A. Makris und K. Tserpes, „A deep learning streaming methodology for trajectory classification,“ *ISPRS International Journal of Geo-Information,* Bd. 10, p. 250, 2021.

[21] Z. Shi, M. Xu, Q. Pan, B. Yan und H. Zhang, „LSTM-based flight trajectory prediction," in *2018 International joint conference on neural networks (IJCNN)*, 2018.

[22] F. Tchuente, N. Baddour und E. D. Lemaire, „Classification of aggressive movements using smartwatches," *Sensors,* Bd. 20, p. 6377, 2020.

[23] J. Siebert, L. Joeckel, J. Heidrich, K. Nakamichi, K. Ohashi, I. Namba, R. Yamamoto und M. Aoyama, „Towards guidelines for assessing qualities of machine learning systems," in *Quality of Information and Communications Technology: 13th International Conference, QUATIC 2020, Faro, Portugal, September 9–11, 2020, Proceedings 13*, 2020.

[24] C. C. Bennett, M. K. Ross, E. Baek, D. Kim und A. D. Leow, „Smartphone accelerometer data as a proxy for clinical data in modeling of bipolar disorder symptom trajectory," *NPJ Digital Medicine,* Bd. 5, p. 181, 2022.

[25] Deutsches Zentrum für Luft- und Raumfahrt e. V., „Cessna 208B Grand Caravan," Deutsches Zentrum für Luft- und Raumfahrt e. V., o. J.. [Online]. Available: https://www.dlr.de/de/forschung-und-transfer/forschungsinfrastruktur/dlr-forschungsflotte/cessna-208b-grand-caravan-1. [Zugriff am 25 03 2024].

[26] C. Mallaun, „Die Basismessanlage der DLR Cessna Grand Caravan C208B (D-FDLR)," Deutsches Zentrum für Luft- und Raumfahrt e. V. [interne Quelle], Braunschweig, 2023.

[27] C.-C. Rossow, K. Wolf und P. Horst, Handbuch der Luftfahrzeugtechnik, Carl Hanser Verlag GmbH Co KG, 2014.

[28] R. Brockhaus, W. Alles und R. Luckner, Flugregelung, Berlin, Heidelberg: Springer, 2013.

[29] P. Vörsmann, Flugmechanik 2 - Flugeigenschaften der Längs- und Seitenbewegung, Braunschweig: Technische Universität Carolo-Wilhelmina Braunschweig, 2005.

[30] C. Raab und S. Burwitz, „DLR Flight Test Program - Uni_Summer_School Luftfahrt 2019," Deutsches Zentrum für Luft- und Raumfahrt e. V. [interne Quelle], Braunschweig, 2019.

[31] Deutsches Zentrum für Luft- und Raumfahrt e.V., „Luftfahrtforschung im DLR - Programm und Strategie," Deutsches Zentrum für Luft- und Raumfahrt e.V., [Online].

Available: https://www.dlr.de/de/forschung-und-transfer/luftfahrt/programm-und-strategie. [Zugriff am 10 12 2023].

[32] H.-C. Oelker, T. Cox, J. Duncan, M. Fırat und E. Topbaş, STO AGARDograph 300 Flight Test Technique Series - Volume 33, National Atlantic Treaty Organisation, 2021.

[33] K. D. Wichman, J. W. Pahle, C. Bahm, J. B. Davidson, B. J. Bacon, P. C. Murphy, A. J. Ostroff und K. D. Hoffler, „High-alpha handling qualities flight research on the nasa f/a-18 high alpha research vehicle," in *NASA Langley High-Angle-of-Attack Conference*, 1996.

[34] National Archives and Records Administration, „Title 14 of the CFR," National Archives and Records Administration, College Park, o. J..

[35] J. Albright, „Departure Obstacle Avoidance," Code 7700, o. J.. [Online]. Available: https://code7700.com/doa_aircraft_performance.htm. [Zugriff am 15 04 2024].

[36] U.S. Naval Test Pilot School, FIXED WING STABILITY AND CONTROL - Theory and Flight Test Techniques, Patuxent River: U.S. Naval Test Pilot School, 1997.

[37] J. P. How, „Aircraft Stability and Control," 2004. [Online]. Available: https://ocw.mit.edu/courses/16-333-aircraft-stability-and-control-fall-2004/resources/lecture_8/. [Zugriff am 10 03 2024].

[38] IBM, „Was ist Deep Learning?," IBM, o. J.. [Online]. Available: https://www.ibm.com/de-de/topics/deep-learning. [Zugriff am 16 03 2024].

[39] P. H. Dang, P. N. Tran, S. Alam und V. N. Duong, „A machine learning-based framework for aircraft maneuver detection and classification," 2021.

[40] IBM, „Was sind neuronale Netze?," IBM, o. J.. [Online]. Available: https://www.ibm.com/de-de/topics/neural-networks. [Zugriff am 14 06 2024].

[41] W. Ertel und N. T. Black, Grundkurs Künstliche Intelligenz, Bd. 4, Wiesbaden: Springer, 2016.

[42] S. J. Russell und P. Norvig, Artificial intelligence a modern approach, London: Pearson, 2010.

[43] K. Pykes, „Cross-Entropy Loss Function in Machine Learning: Enhancing Model Accuracy," DataCamp, 01 2024. [Online]. Available:

https://www.datacamp.com/tutorial/the-cross-entropy-loss-function-in-machine-learning. [Zugriff am 22 06 2024].

[44] D. P. Kingma und J. Ba, „Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.

[45] V. Efimov, „Understanding Deep Learning Optimizers: Momentum, AdaGrad, RMSProp & Adam," Medium, 30 12 2023. [Online]. Available: https://towardsdatascience.com/understanding-deep-learning-optimizers-momentum-adagrad-rmsprop-adam-e311e377e9c2. [Zugriff am 06 07 2024].

[46] M. Grandini, E. Bagli und G. Visani, „Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756,* 2020.

[47] V. Boreiko und M. Augustin, „Die Blackbox des Deep Learning bei der Bildklassifizierung öffnen," Eberhard Karls Universität Tübingen, 24 01 2023. [Online]. Available: https://www.machinelearningforscience.de/die-blackbox-des-deep-learning-bei-der-bildklassifizierung-oeffnen/. [Zugriff am 25 06 2024].

[48] A. Ng, *Basic Recipe for Machine Learning (C2W1L03),* DeepLearningAI, Hrsg., YouTube, 2017.

[49] N. Lang, „Was ist Overfitting?," Niklas Lang, 10 12 2022. [Online]. Available: https://databasecamp.de/ki/overfitting. [Zugriff am 26 06 2024].

[50] A. Ng, *Bias/Variance (C2W1L02),* DeepLearningAI, Hrsg., YouTube, 2017.

[51] PyTorch Lightning, „EARLY STOPPING," PyTorch Lightning, o. J.. [Online]. Available: https://lightning.ai/docs/pytorch/stable/common/early_stopping.html. [Zugriff am 26 06 2024].

[52] N. Lang, „Was ist Underfitting?," Niklas Lang, 13 09 2023. [Online]. Available: https://databasecamp.de/ki/underfitting. [Zugriff am ß7 ß7 2024].

[53] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever und R. Salakhutdinov, „Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* Bd. 15, p. 1929–1958, 2014.

[54] R. Nirthika, S. Manivannan, A. Ramanan und R. Wang, „Pooling in convolutional neural networks for medical image analysis: a survey and an empirical study," *Neural Computing and Applications,* Bd. 34, p. 5321–5347, 2022.

[55] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard und L. D. Jackel, „Backpropagation applied to handwritten zip code recognition," *Neural computation,* Bd. 1, p. 541–551, 1989.

[56] A. Krizhevsky, I. Sutskever und G. E. Hinton, „Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems,* Bd. 25, 2012.

[57] Z. Wang, W. Yan und T. Oates, „Time series classification from scratch with deep neural networks: A strong baseline," *2017 International Joint Conference on Neural Networks (IJCNN),* pp. 1578-1585, 2016.

[58] M. Cochez, *Convolutional Neural Networks,* V. U. Amsterdam, Hrsg., Amsterdam, 2020.

[59] K. He, X. Zhang, S. Ren und J. Sun, „Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[60] GeeksforGeeks, „Vanishing and Exploding Gradients Problems in Deep Learning," GeeksforGeeks, 13 12 2023. [Online]. Available: https://www.geeksforgeeks.org/vanishing-and-exploding-gradients-problems-in-deep-learning/. [Zugriff am 06 06 2024].

[61] GeeksforGeeks, „Introduction to Recurrent Neural Network," GeeksforGeeks, 05 06 2024. [Online]. Available: https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/. [Zugriff am 05 06 2024].

[62] S. Hochreiter und J. Schmidhuber, „Long short-term memory," *Neural computation,* Bd. 9, p. 1735–1780, 1997.

[63] E. Arts, A. Kamtsiuris, H. Meyer, F. Raddatz, A. Peters und S. Wermter, „Trajectory Based Flight Phase Identification with Machine Learning for Digital Twins," 2022.

[64] A. Zhang, Z. C. Lipton, M. Li und A. J. Smola, Dive into Deep Learning, Cambridge University Press, 2023.

[65] R. Tan, „LSTMs Explained: A Complete, Technically Accurate, Conceptual Guide with Keras," Medium, 02 09 2020. [Online]. Available: https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2. [Zugriff am 17 06 2024].

[66] H. Meyer, J. Zimdahl, A. Kamtsiuris, R. Meissner, F. Raddatz, S. Haufe und M. Bäßler, „Development of a digital twin for aviation research," 2020.

[67] D. Kreuzberger, N. Kühl und S. Hirschl, *Machine Learning Operations (MLOps): Overview, Definition, and Architecture,* 2022.

[68] REFA Bundesverband e.V., „Paarweiser Vergleich," in *Industrial Engineering Standardmethoden zur Produktivitätssteigerung*, Darmstadt, REFA Bundesverband e.V., 2015, pp. 103-108.

[69] University Corporation of Atmospheric Research, „Network Common Data Form (NetCDF)," University Corporation of Atmospheric Research, o. J.. [Online]. Available: https://www.unidata.ucar.edu/software/netcdf/. [Zugriff am 04 16 2024].

[70] J. Frochte, Maschinelles Lernen: Grundlagen und Algorithmen in Python, Carl Hanser Verlag GmbH Co KG, 2020.

[71] T. Hashemi, „Feature Selection für Machine Learning - Predictive Analytics - Teil 1," 12 03 2018. [Online]. Available: https://tareq.de/Feature-Selection-Predictive-Analytics-Teil1/. [Zugriff am 23 06 2024].

[72] J. Brownlee, „1D Convolutional Neural Network Models for Human Activity Recognition," Guiding Tech Media, 28 08 2020. [Online]. Available: https://machinelearningmastery.com/cnn-models-for-human-activity-recognition-time-series-classification/. [Zugriff am 21 05 2024].

[73] W. Folta, „Supervised learning: setting labels on sliding windows of sensor data," Stack Exchange Inc, 13 12 2017. [Online]. Available: https://stats.stackexchange.com/questions/318240/supervised-learning-setting-labels-on-sliding-windows-of-sensor-data. [Zugriff am 05 07 2024].

[74] N. C. A. de Freitas, T. L. C. da Silva, J. A. F. de Macêdo, L. M. Junior und M. G. Cordeiro, „Using deep learning for trajectory classification," in *Proceedings of the 13th International Conference on Agents and Artificial Intelligence (ICAART 2021)*, 2021.

[75] A. Oppermann, „Optimierung in Deep Learning: AdaGrad, RMSProp, ADAM," o. J.. [Online]. Available: https://artemoppermann.com/de/optimierung-in-deep-learning-adagrad-rmsprop-adam/. [Zugriff am 24 06 2024].

[76] I. Goodfellow, Y. Bengio und A. Courville, Deep Learning, MIT Press, 2016.

[77] Optuna Contributors, „Efficient Optimization Algorithms,“ Optuna Contributors, 2018. [Online]. Available: https://optuna.readthedocs.io/en/stable/tutorial/10_key_features/003_efficient_optimization_algorithms.html. [Zugriff am 23 06 2024].

[78] Run:ai, „Hyperparameter Tuning - Top 5 Techniques,“ Run:ai, [Online]. Available: https://www.run.ai/guides/hyperparameter-tuning. [Zugriff am 08 06 2024].

[79] S. M. LaValle, M. S. Branicky und S. R. Lindemann, „On the relationship between classical grid search and probabilistic roadmaps,“ *The International Journal of Robotics Research,* Bd. 23, p. 673–692, 2004.

[80] J. Bergstra und Y. Bengio, „Random search for hyper-parameter optimization.,“ *Journal of machine learning research,* Bd. 13, 2012.

[81] W. Shao, A. Prabowo, S. Zhao, P. Koniusz und F. D. Salim, „Predicting flight delay with spatio-temporal trajectory convolutional network and airport situational awareness map,“ *Neurocomputing,* Bd. 472, p. 280–293, 2022.

[82] H. Rakhshani, H. I. Fawaz, L. Idoumghar, G. Forestier, J. Lepagnot, J. Weber, M. Brévilliers und P.-A. Muller, „Neural architecture search for time series classification,“ in *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020.

[83] P. Lara-Benítez, M. Carranza-García, J. M. Luna-Romera und J. C. Riquelme, „Temporal convolutional networks applied to energy-related time series forecasting,“ *applied sciences,* Bd. 10, p. 2322, 2020.

[84] G. Youness, N. U. T. Phan und B. C. Boulakia, „BootBOGS: Hands-on optimizing Grid Search in hyperparameter tuning of MLP,“ in *AICCSA 2023: 20th ACS/IEEE International Conference on Computer Systems and Applications*, 2023.

[85] R. Marco, S. S. S. Ahmad und S. Ahmad, „An Improving Long Short Term Memory-Grid Search Based Deep Learning Neural Network for Software Effort Estimation.,“ *International Journal of Intelligent Engineering & Systems,* Bd. 16, 2023.

[86] AIME GmbH, „Effizientes Deep Learning: Training des ResNet50 Models auf dem ImageNet-Datensatz,“ AIME GmbH, 2024. [Online]. Available: https://www.aime.info/blog/de/de-resnet50-training-with-imagenet/. [Zugriff am 23 06 2024].

[87] J. Nabi, „Hyper-parameter Tuning Techniques in Deep Learning,“ Medium, 16 03 2019. [Online]. Available: https://towardsdatascience.com/hyper-parameter-tuning-techniques-in-deep-learning-4dad592c63c8. [Zugriff am 07 06 2024].

[88] C. Molnar, Interpretable Machine Learning - A Guide for Making Black Box Models Explainable, Unabhängig veröffentlicht, 2024.

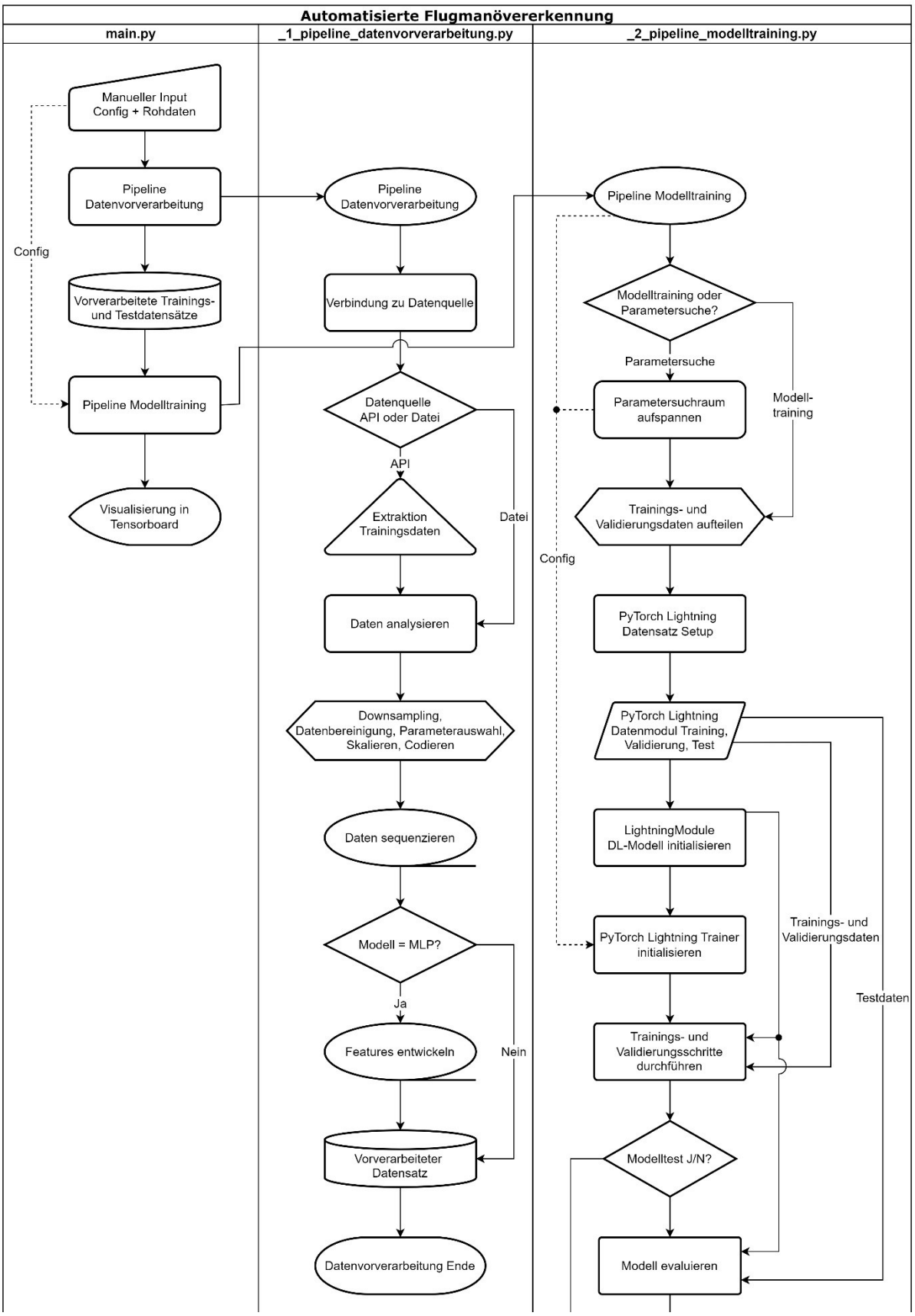
[89] I. Charabi, „Supervised learning: setting labels on sliding windows of sensor data,“ Stack Exchange Inc, 24 05 2024. [Online]. Available: https://stats.stackexchange.com/questions/318240/supervised-learning-setting-labels-on-sliding-windows-of-sensor-data. [Zugriff am 24 06 2024].

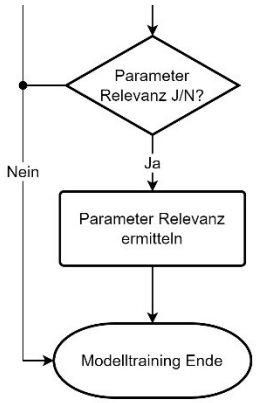
A 1 Signalliste Cessna Grand Caravan 208B

Channel_List_FDLR_2020		
Channel Name	**Description**	**Unit**
ABSHUM	Absolute Luftfeuchtigkeit	g/m³
ALPHA	Anstellwinkel	deg
BETA	Schiebewinkel	deg
CAS	Berechnete Fluggeschwindigkeit	m/s
CPT_F	Kraftsensor für Steuerhorn Pilot	N
CPT_L	Wegsensor für Steuerhorn Copilot	mm
ELEVATOR	Höhenruderwinkel	deg
EWV	IGI Ost West Geschwindigkeit ohne Korrektur	m/s
H	Flughöhe über Meeresspiegel	m
HP	Druckhöhe	m
IAS	Angezeigte Fluggeschwindigkeit	m/s
IGI_RMSX	IGI Referenz Position x	m
IGI_RMSY	IGI Referenz Position y	m
IGI_RMSZ	IGI Referenz Position z	m
IRS_ALT	IGI Flughöhe über WGS84 Ellipsoid	m
IRS_ATA	IGI Wahrer Kurs	deg
IRS_AXB	IGI Beschleunigung entlang der x-Achse	m/s²
IRS_AYB	IGI Beschleunigung entlang der y-Achse	m/s²
IRS_AZB	IGI Beschleunigung entlang der z-Achse	m/s²
IRS_AZG	IGI Vertikalbeschleunigung	m/s²
IRS_EWV	IGI Ost West Geschwindigkeit	m/s
IRS_GS	IGI Fluggeschwindigkeit über Grund	m/s
IRS_HDG	IGI Wahrer Gierwinkel	deg
IRS_LAT	IGI Breitengrad Position über WGS84 Ellipsoid	deg
IRS_LON	IGI Längengrad Position über WGS84 Ellipsoid	deg
IRS_NSV	IGI Nord Süd Geschwindigkeit	m/s
IRS_P	IGI Rollgeschwindigkeit bzw. Rollrate	deg/s
IRS_PHI	IGI Rollwinkel	deg
IRS_Q	IGI Nickgeschwindigkeit bzw. Nickrate	deg/s
IRS_R	IGI Giergeschwindigkeit bzw. Gierrate	deg/s
IRS_THE	IGI Nickwinkel	deg
IRS_VV	IGI Steiggeschwindigkeit	m/s
MC	Machzahl	Ma
MFL_FF	Durchflussrate Kraftstoff	kg/h
MFL_FOB	Kraftstoff an Bord	kg
MFL_FU	Verbrauchter Kraftstoff	kg
MFL_UNT	Kraftstoffverbrauch	-
MIXRATIO	Kraftstoff-Luft-Gemisch	g/kg

NSV	IGI Nord Süd Geschwindigkeit	m/s
PS	Statischer Druck	hPa
QC	Dynamischer Druck	hPa
RELHUM	Relative Luftfeuchtigkeit	%
SOURCE	Datenquelle Feuchtigkeit (1:Vaisala/2:Iy_a(lin)/3:Iy_a(log)/4:GE-dewpoint),	-
TAS	Wahre Fluggeschwindigkeit	m/s
TAT	Gesamttemperatur der Luft (Enteisung korrigiert, von BDY-TTQ)	K
TD	Temperatur des Kondensationspunkts	K
TERRAINHEIGHT	Flughöhe über Grund	m
THETA	Potentielle Temperatur	K
THETA_V	Virtuelle Potentielle Temperatur	K
sys_time	Sekunden nach Mitternacht (UTC-Zeitformat)	s
TS	Statische Temperatur der Luft (von BDY-TTQ)	K
TV	Virtuelle Temperatur	K
U	Windvektor Ost-Komponente	m/s
V	Windvektor Nord-Komponente	m/s
VV	Steiggeschwindigkeit (von IRS)	m/s
W	Windvektor vertikale Komponente	m/s
WA	Horizontale Windrichtung	deg
WS	Horizontale Windgeschwindigkeit	m/s

A 2 Flussdiagramm Python-Modul




Parameter Relevanz J/N?
Ja
Nein
Parameter Relevanz ermitteln
Modelltraining Ende

A 3 Konfiguration Hyperparameter Python-Code main.py

```python
    hyperparameter_defaults = dict(
        # Data extraction and loading
        data_source_type="file",
        data_source_path=r"H:\flightmanueverrecognition\data\Rohdaten\FDLR_train_flights.pkl",

        # Data cleaning and preprocessing
        downsample_data=True,
        missing_values_strategy='drop_cols',
        scaling_strategy='standard',
        encoding_strategy='integer',
        window_data=True,
        window_size=32,
        overlap=0.5,
        label_strategy='last_value',
        test_ingestion=False,
        feature_store_path=r'H:\flightmanueverrecognition\data\FeatureStore',

        # Data preparation and validation
        val_size=0.2,

        # Model training
        mode='train_model',
        evaluate_model=False,
        feature_subset=None,
        model='lstm',
        checkpoint_available=False,
        model_checkpoint_path=None,
        early_stopping=False,
        num_epochs=1,
        batch_size=16,
        save_top_k=1,
        accelerator="gpu" if torch.cuda.is_available() else "cpu",
        learning_rate=0.001,
        num_layers=2,
        hidden_size=50,  # hidden_size for LSTM
        # hidden_size=[64, 128, 128],  # hidden_size for MLP
        # hidden_size=[64, 128, 128],  # hidden_size for ResNet
        dropout=0.2,
    )
```