UNIVERSITÄT
LEIPZIG

Institut für Informatik
Fakultät für Mathematik und Informatik
Abteilung Datenbanken

Entwicklung eines Privatsphäre-erhaltenden Transformers zur Stresserkennung anhand von Smartwatch-Health-Daten

Masterarbeit

vorgelegt von:
Borislav Degenkolb

Matrikelnummer:
3726365

Betreuer:
Prof. Dr. Erhard Rahm
Lucas Lange

Zusammenfassung

Stress ist ein weit verbreitetes Phänomen, das nicht nur zu psychischen, sondern auch zu physischen Erkrankungen führen kann. Die frühzeitige Erkennung von Stress mittels Machine-Learning-Verfahren ermöglicht es, zur Stressbewältigung und -reduktion beizutragen und somit die Lebensqualität der betroffenen Personen zu erhöhen.

Im Rahmen dieser Masterarbeit wurde ein Ansatz zur Stresserkennung basierend auf einer Time-Series-Classification-Transformer-Architektur (TSCT) untersucht und die Auswirkungen von Differential Privacy (DP) auf die Modellleistung evaluiert. Die Ergebnisse zeigten, dass der Einsatz von TSCTs eine vielversprechende Möglichkeit darstellt, Stress im WESAD-Datensatz mit hoher Genauigkeit zu erkennen. Das Modell erreichte eine durchschnittliche Genauigkeit von 91,89 Prozent und einen durchschnittlichen F1-Score von 91,61 Prozent. Die Anwendung von DP ermöglichte den Schutz der Privatsphäre der Teilnehmer, führte jedoch zu Einbußen in der Modellleistung, insbesondere im Hinblick auf die Präzision. Bei den besten trainierten Modellen mit DP waren Genauigkeitseinbußen von etwa 4 Prozent und ein F1-Score-Abfall von etwa 25 Prozentpunkten zu verzeichnen. Die Arbeit weist zusätzlich einige Limitationen auf, wie die mangelnde Generalisierbarkeit der Ergebnisse, die Interpretierbarkeit von Transformer-Modellen und die praktische Anwendbarkeit des Ansatzes in realen Anwendungen.

Zusammenfassend zeigt diese Masterarbeit das Potenzial von TSCTs für die Stresserkennung und die Anwendung von DP zur Wahrung der Privatsphäre, obwohl die Performance von privaten Modellen und der Zusammenhang zur Epsilon-Metrik schwer abzuschätzen sind.

Abstract

Stress is a widespread phenomenon that can lead not only to psychological but also to physical illnesses. Early detection of stress using machine learning techniques enables contributions to stress management and reduction, thereby improving the quality of life for affected individuals.

In the context of this master's thesis, an approach to stress detection based on a Time-Series Classification Transformer architecture (TSCT) was investigated, while the effects of Differential Privacy (DP) on model performance were evaluated. The results showed that the use of TSCTs is a promising way to detect stress in the WESAD dataset with high accuracy. The model achieved an average accuracy of 91.89 percent and an average F1 score of 91.61 percent. The application of DP allowed the protection of participants privacy but resulted in compromises in model performance, particularly in terms of precision. For the best-trained models with DP, accuracy losses of about 4 percent and an F1 score drop of approximately 25 percentage points were observed. The work also has some limitations, such as the lack of generalizability of the results, the interpretability of transformer models, and the practical applicability of the approach in real-world applications.

In summary, this master's thesis demonstrates the potential of TSCTs for stress detection and the application of DP for maintaining privacy, although the performance of private models and the relationship to the epsilon metric are difficult to assess.

Borislav Degenkolb
3726365

Inhaltsverzeichnis

Abkürzungsverzeichnis

BVP	Blut-Volumen-Puls
CNN	Convolutional Neural Network
$DP - SG$	differentiell privater stochastischer Gradientenabstieg
DP	Differential Privacy
EDA	elektrodermale Aktivität
EEG	Elektroenzephalogramm
EKG	Elektrokardiogramm
EMG	Elektromyographie
FFT	Fast-Fourier-Transformation
GAN	Generative Adversarial Networks
$LOSO$	Leave-One-Subject-Out
$LSTM$	Long Short-Term Memory
ML	Machine Learning
MLP	Multi-Layer-Perceptron
NLP	Natural Language Processing
PCA	Principal Component Analysis
RNN	Rekurrentes neuronales Netz
$TSCT$	Time-Series-Classification-Transformer
$WESAD$	Wearable Stress and Affect Detection

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

1.1. Motivation

Jeder intrinsische oder extrinsische Stimulus, der eine biologische Reaktion hervorruft, wird als Stress bezeichnet. Die kompensatorischen Reaktionen auf diese Belastungen werden als Stressreaktionen definiert. Je nach Art, Zeitpunkt und Schwere des angewendeten Reizes kann Stress verschiedene Wirkungen auf den Körper ausüben, die von Veränderungen der Homöostase bis hin zu lebensbedrohlichen Auswirkungen und Tod reichen. In vielen Fällen sind die pathophysiologischen Komplikationen von Krankheiten auf Stress zurückzuführen [1], dabei sind Auswirkungen auf das Immunsystem [2], das Entstehen von Krebs [3], Herz-Kreislauf-Erkrankung [4], Depression [5] oder auch Diabetes [6, 7] erforscht.

Zunehmend wird anerkannt, dass Arbeitgeber in vielen Fällen gesetzlich verpflichtet sind, dafür zu sorgen, dass Arbeitnehmer nicht krank werden. Es liegt auch in ihrem langfristigen wirtschaftlichen Interesse Stress im Unternehmen zu vermeiden, da Stress voraussichtlich zu einer hohen Personalfluktuation, einem Anstieg der Krankheitsabwesenheit und Frühverrentung, zu verminderter Arbeitsleistung sowie geringerer Kundenzufriedenheit führt. [8]

Demnach ist es nicht verwunderlich, dass (frühzeitige) Erkennung von Stress in der Forschung immer mehr an Bedeutung gewinnt. Klassische Anzeichen für Stress lassen sich am Verhalten von Menschen erkennen, insbesondere an Verhaltensänderungen [8]. Akute Reaktionen auf Stress können sich in den Bereichen Gefühle (z. B. Angst, Depression), Verhalten (z. B. zurückgezogen, aggressiv), Denken (z. B. Konzentrations- und Problemlösungsschwierigkeiten) oder körperliche Symptome (z. B. Herzklopfen, Übelkeit, Kopfschmerzen) zeigen.

Technische Verfahren hingegen stützen sich auf das Interpretieren von Biosignaldaten. Zu den Biosignalen, die im Zusammenhang mit Stressoren zuverlässig gemessen werden können, gehören unteranderem physiologische (EEG, EKG, EDA, EMG) und physische Messgrößen (Atemfrequenz, Sprache, Hauttemperatur, Pupillengröße, Augenaktivität) [9].

Die schnelle Entwicklung von Wearable-Devices und die weite Verbreitung smarter Geräte erleichtern die Aufzeichnung von Biosignalen während täglicher Aktivitäten [10]. Diese Technologien ermöglichen es, kostengünstige Stressüberwachungssysteme, welche auf tragbaren Sensoren basieren, zu entwerfen.

Die vorliegende Arbeit greift die Erkenntnisse von Gil-Martin et al. [11] auf. Statt dem Einsatz eines Convolutional Neural Networks zur Klassifizierung der Sensordaten aus dem WESAD Datensatz [12] in die Stadien „Stress“ und „kein Stress“ wird in dieser Arbeit der Ansatz über eine Transformer-Architektur [13] ausprobiert und im Anschluss verglichen. Da Biosignale zudem personenbezogene Daten sind, wird zusätzlich geprüft, inwieweit die Daten auch nach einem Anonymisierungsverfahren für die angewandte Methode nutzbar sind.

1.2. Fragestellung

1. Welche Transformer-Architektur eignet sich am besten zur Klassifizierung / Erkennung von Stress aus dem WESAD Datensatz?
2. Wie gut, auch im Vergleich zu den Ergebnissen von Gil-Martin et al. [11], sind die Ergebnisse dieser Vorgehensweise?
3. Wie groß sind die Einbußen in den Ergebnissen, wenn der Datensatz zuvor mit Hilfe von Differential Privacy anonymisiert wird?

1.3. Aufbau der Arbeit

Nachdem die Relevanz der frühzeitigen Stresserkennung verdeutlicht wurde, ist folgende Arbeit entsprechend aufgebaut:

- In **Kapitel 2** werden die für das Verständnis der Arbeit nötigen **Grundlagen** näher erläutert. Dabei wird zunächst auf die Stresserkennung in Smartwatch-Daten im Allgemeinen Bezug genommen. Weiterführend werden technische Grundlagen, wie die genutzte Transformer-Architektur und das privatsphäre-erhaltende Framework der Differential Privacy, erklärt. Da zur Evaluation des trainierten Modells die LOSO-Evaluation genutzt wurde, wird diese zum Ende des Kapitels kurz beschrieben.
- In **Kapitel 3**, den **verwandten Arbeiten,** findet sich eine Übersicht der Ergebnisse aus aktuellen Forschungsarbeiten zum angewendeten Datensatz sowie der Transformer-Architektur und Differential Privacy.
- **Kapitel 4** beschreibt die **Methode**, mit der die neue Transformer-Architektur an eine vorgegebene Signalvorverarbeitung adaptiert und schlussendlich evaluiert wurde. Zusätzlich wird das Vorgehen des Einsatzes von Differential Privacy und der experimentelle Aufbau näher beleuchtet.
- In **Kapitel 5** werden Auszüge der **Implementierung** aufgezeigt und ausgearbeitet. Dabei wird neben der eigenen Adaptierung des Transformers auch auf dessen Quelle aus Keras sowie die vorgegebene Entwicklung der Signalvorverarbeitung Bezug genommen
- Die **Evaluation und Ergebnisse** der folgenden Arbeit finden sich in **Kapitel 6**. Hier werden sowohl Performance- aber auch Hyperparametertuning-Parameter tabellarisch und graphisch aufgezeigt
- **Diskutiert** werden die Ergebnisse und das Vorgehen der Arbeit in **Kapitel 7**.
- Das letzte **Kapitel 8** enthält das **Fazit** der Ausarbeitung. Außerdem wird hier ein **Ausblick** gegeben, in welche Forschungsrichtung mit den Ergebnissen weitergearbeitet werden kann.

2. Grundlagen

2.1. Stresserkennung anhand von Biosignalen

Stress ist eine natürliche Reaktion des Körpers auf Belastungen, welche eine Anpassung an veränderte Umstände ermöglicht [14]. Die Stressreaktion wird dabei von verschiedenen Hormonen und Neurotransmittern, wie z.B. Adrenalin, Noradrenalin und Cortisol, reguliert. Akuter Stress stellt eine kurzfristige Stressreaktion dar. Er wird von einer schnellen Freisetzung von Hormonen begleitet, welche die Herzfrequenz, den Blutdruck und die Atemfrequenz erhöhen und die Energiereserven des Körpers mobilisieren. Chronischer Stress dagegen ist eine langfristige Belastung, die eine kontinuierliche Aktivierung des Stresssystems nach sich zieht und zu einer Erschöpfung der Energiereserven führen kann [15].

Um Stress zu erkennen, bedarf es verschiedener physiologischer Messmethoden. Dazu gehören Messungen des Pulses mit Hilfe eines Elektrokardiogramm (EKG), der elektrodermalen Aktivität (EDA), der Muskelaktivität, der Atmung, der Temperatur sowie des Hormonspiegels im Blut [16].

Die Herzfrequenz ist das am weitesten verbreitete und einfachste Maß zur Einschätzung des Stressniveaus. Alternativ kann auch das mittlere RR-Intervall, das heißt das Intervall zwischen aufeinanderfolgenden Herzschlägen, verwendet werden, das in einem umgekehrten Verhältnis zur Herzfrequenz steht. Vielen Studien belegen, dass die Herzfrequenz in Stresssituationen deutlich ansteigt [17, 18, 19].

Die EDA misst die Aktivität der Schweißdrüsen und ist eng mit dem sympathischen Nervensystem verbunden. Eine erhöhte EDA wird häufig mit emotionaler Erregung und Stress assoziiert [20].

Die Temperaturmessung kann zeigen, ob der Körper aufgrund von Stress eine Veränderung der Hauttemperatur aufweist. Hormonelle Messungen, wie die Bestimmung des Cortisolspiegels im Blut, können ebenfalls Aufschluss über den Stresslevel geben. Die Kombination mehrerer Messmethoden führt oft zu einer höheren Genauigkeit bei der Stresserkennung. Allerdings gibt es auch Herausforderungen bei der Interpretation der Messwerte, da sie von individuellen Faktoren, wie zum Beispiel Alter, Geschlecht und Fitness, beeinflusst werden können. [21]

2.2. Time-Series-Classification-Transformer

Die Verwendung von Machine-Learning-Modellen zur Klassifizierung von Biosignalen wie z.B. Elektrokardiogrammen (EKG), Elektroenzephalographien (EEG) und Pulsoximeterdaten hat in den letzten Jahren zunehmend an Bedeutung gewonnen. Eine Möglichkeit, diese Art von Daten zu verarbeiten, ist die Verwendung von Time-Series-Transformer-Architekturen.

Traditionelle Transformer sind tiefe neuronale Netze, die ursprünglich für die Verarbeitung von natürlicher Sprache entwickelt wurden [13]. Sie basieren auf der Verwendung von Attention-Mechanismen,

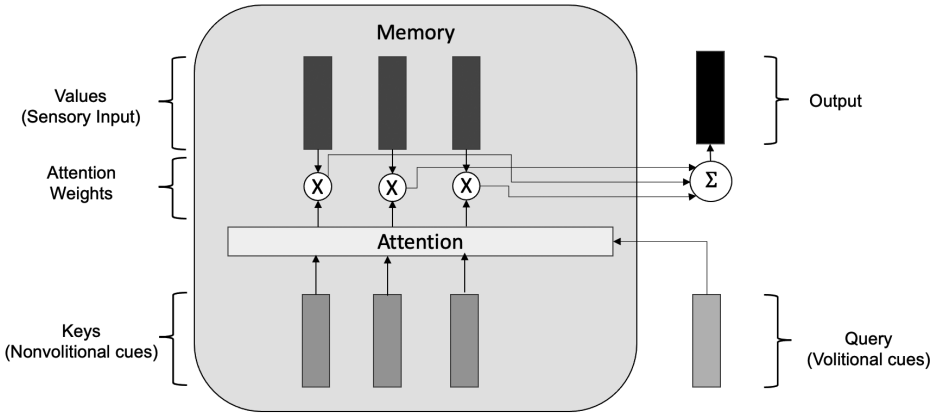


Abbildung 2.2.: Attention-Mechanismus, der die Interaktionen zwischen Abfragen (*querys*), Schlüssel (*keys*) und Werten (*values*) aufweist [23]

die es ermöglichen, bestimmte Teile der Eingabedaten zu betonen und andere hingegen zu ignorieren. Aus diesem Grund sind sie besonders gut geeignet für die Verarbeitung von sequentiellen Daten wie z.B. Zeitreihen.

Wie in Abbildung 2.1 zu sehen ist, besteht die Architektur aus mehreren Schichten von Self-Attention-Blöcken, die parallel zueinander arbeiten. In jedem Block gibt es drei Hauptkomponenten: Multi-Head-Attention, Feedforward-Netzwerk und Layer-Normalisierung.

Multi-Head-Attention ist die wichtigste Komponente des Transformer-Modells. Sie erlaubt dem Modell, sich auf verschiedene Teile der Eingabe-Sequenz zu konzentrieren und sie in der Darstellung der Eingabe zu berücksichtigen. Wie in Abbildung 2.2 gezeigt, kann der Aufmerksamkeitsmechanismus als ein Speicher mit Schlüsseln und Werten und einer zusätzlichen Schicht betrachtet werden. Diese Schicht, wenn abgefragt, erzeugt eine Ausgabe aus dem Wert, dessen Schlüssel die Eingabe abbildet [22].

Das Feedforward-Netzwerk besteht aus zwei linearen Transformationen, bei denen eine nicht-sigmoidale, gleichgerichtete, lineare Einheit ($ReLU$)-Aktivierung dazwischenliegt. Es wird angewendet, um die Eingabe-Sequenz zu transformieren, nachdem sie durch den Multi-Head-Attention-Block verarbeitet wurde. Die

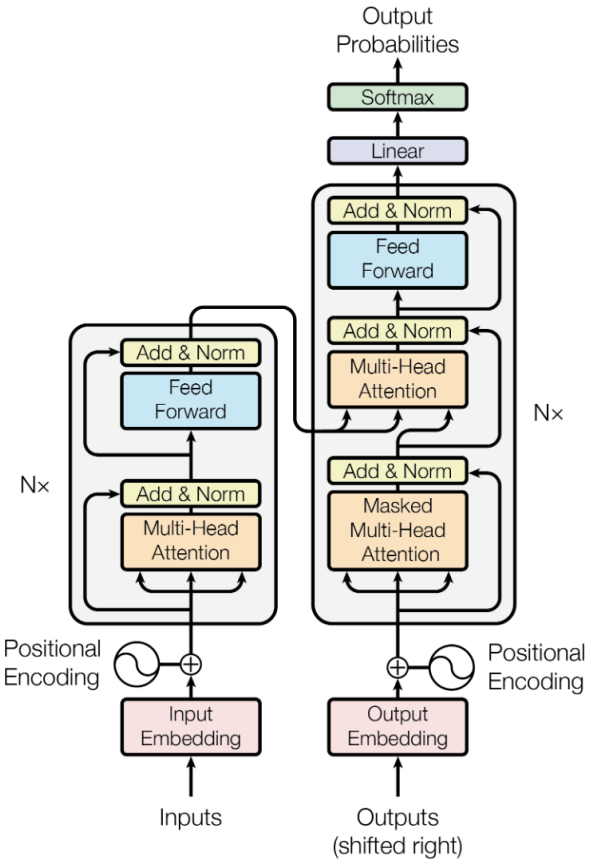


Abbildung 2.1.: Transformer Architektur [13]

Layer-Normalisierung normalisiert die Ausgabe
jeder Schicht, um den Trainingsprozess zu stabilisieren und das Modell zu beschleunigen.

Die Besonderheit bei Time-Series-Classification-Transformern ist, dass diese, im Gegensatz zu traditionellen Transformern, keinen Decoder-Teil besitzen, da sie keine Ausgabesequenz generieren (vgl. [24, 25, 26]). In einem Time-Series-Classification-Transformer besteht die Eingabe aus einer Sequenz von Merkmalsvektoren, die auf diskrete Zeitpunkte innerhalb einer Zeitreihe bezogen sind. Jeder Merkmalsvektor repräsentiert die Daten an einem bestimmten Zeitpunkt in der Zeitreihe. Der Transformer verwendet dann die Aufmerksamkeitsmechanismen, um die Beziehungen zwischen den Merkmalsvektoren in der Zeitreihe zu modellieren und eine Vorhersage über die Klassenzugehörigkeit zu treffen [27].

2.3. Differential Privacy

Differential Privacy (DP) ist ein Konzept des Datenschutzes, das in den letzten Jahren zunehmend an Bedeutung gewonnen hat. Es wurde erstmals im Jahr 2008 von Dwork et al. [28] eingeführt. Seitdem wird das Konzept in der Forschung kontinuierlich weiterentwickelt und angewendet.

Differential Privacy ist ein mathematisches Konzept, das es erlaubt, statistische Analysen auf Daten durchzuführen, ohne dass dabei individuelle Informationen der Daten offengelegt werden. Das Ziel von Differential Privacy ist es, die Privatsphäre der Teilnehmenden zu schützen, während gleichzeitig die Nützlichkeit der Daten für die Analyse beibehalten wird.

Das Konzept wird durch eine mathematische Funktion definiert, die als *Differential* bezeichnet wird. Diese Funktion ermöglicht es, zu messen, wie sehr sich das Ergebnis einer statistischen Analyse verändert, wenn ein einzelner Datensatz hinzugefügt oder entfernt wird. Ein hohes Differential bedeutet dabei, dass das Ergebnis sehr sensibel auf individuelle Daten reagiert, während das Ergebnis bei einem niedrigen Differential kaum beeinflusst wird. Diese Funktion kann beispielsweise folgendermaßen aussehen:

$$d(x) = max(||f(D) - f(D')||) \tag{2.1}$$

wobei D und D' Datensätze sind, die sich nur in einem einzelnen Eintrag unterscheiden. Um DP zu erreichen, muss die oben beschriebene Differentialfunktion 2.1 unter einer bestimmten Schwelle gehalten werden [28]. Diese Schwelle wird als *Privacy Budget* ε bezeichnet und legt fest, wie viel Veränderung in den Ergebnissen einer statistischen Analyse zulässig ist, bevor die Privatsphäre der Teilnehmenden verletzt wird. Ein höheres ε ermöglicht dabei größere Veränderungen in den Ergebnissen, führt aber auch zu einem geringeren Schutz der Privatsphäre. Ein $\varepsilon = \infty$ weist darauf hin, dass keine DP-Kriterien erfüllt sind. Die Metrik wird begleitet von der Wahrscheinlichkeit, dass Privatsphäre durch zufälliges Durchsickern von Informationen verletzt wird. Diese wird als δ bezeichnet und ist von der Größe des Datensatzes abhängig [29]. Ein Algorithmus A welcher auf einem Datensatz-Set S trainiert wird, ist formal betrachtet (ε, δ)-differentiell-privat, wenn für alle Datensätze D und D' gilt:

$$Pr[A(D) \in S] \leq e^{\varepsilon} Pr[A(D') \in S] + \delta \tag{2.2}$$

Um gegen datenschutzgefährdende Angriffe auf Datensätze ausreichend geschützt zu sein, sollte ein $\varepsilon \leq 1$ und ein $\delta \ll 1/n$ gewählt werden [30]. n ist hierbei die Anzahl der *training samples*.

Ein häufig verwendetes Verfahren zur Erreichung von Differential Privacy ist das Hinzufügen von zufälligem Rauschen [28]. Eine andere Methode besteht darin, aggregierte Daten zu verwenden, anstatt einzelne Datenpunkte zu analysieren.

Differential Privacy findet Anwendung in verschiedenen Bereichen, wie zum Beispiel in der medizinischen Forschung [29, 31, 32, 33, 34, 35], im Wirtschaftssektor [36, 37, 38] und der Analyse von sozialen Netzwerken [39, 40, 41, 42, 43, 44]. Differential Privacy ermöglicht es, wertvolle Informationen aus Daten zu gewinnen, ohne dabei die Privatsphäre der Teilnehmenden zu gefährden.

3. Verwandte Arbeiten

3.1. WESAD-Datensatz

Die in dieser Arbeit verwendeten Daten entstammen dem Paper *Ïntroducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection"* von Schmidt et al. [12]. Der hier vorgestellte multimodale WESAD-Datensatz besteht aus Daten, die zum einen von tragbaren Geräten wie Smartwatches und Brustgurten erfasst wurden, aber auch aus den Daten eines Fragebogens, um Stress und Emotionen von Benutzern zu detektieren. Der Datensatz setzt sich aus folgenden Komponenten zusammen:

- Physiologische Signale: Elektrokardiographische (EKG) Signale, die die elektrischen Aktivitäten des Herzens messen, Hautleitfähigkeitssignale (elektrodermale Aktivität), die Veränderungen in der Schweißproduktion messen, und Atmungssignale, die die Atemfrequenz und Atemtiefe messen.
- Bewegungsdaten: Beschleunigungsmesser- und Gyroskopdaten, die Bewegungen des Körpers erfassen, wie zum Beispiel Schritte, Körperhaltung und Bewegungen.
- Audioaufnahmen: Audioaufnahmen, die während der Datenerfassung gemacht wurden, um Umgebungsgeräusche und möglicherweise auch die Stimme des Benutzers aufzuzeichnen.

Die Daten wurden von 15 gesunden Probanden während der Durchführung von verschiedenen Aktivitäten erfasst, darunter Baseline-Übungen, kognitiven Tests und emotionalen Induktionen. (vgl. Abbildung 3.1) Die Datenerfassung erfolgte sowohl im Labor als auch in der freien Umgebung der Probanden. Das Ergebnis der Arbeit ist zum einen ein vorverarbeiteter und annotierter Datensatz, welcher eine automatische Analyse durch Machine Learning (ML)-Verfahren ermöglicht, und zum anderen die Resultate von angewendeten ML-Verfahren zur Stresserkennung. Diese werden in folgendem Kapitel 3.4 tabellarisch aufgezeigt.

Die in vorliegender Arbeit zur Stresserkennung genutzten Daten stammen aus der Smartwatch *Empatica E4*, welche den Blut-Volumen-Puls (BVP), elektrodermische Aktivität (EDA), Hauttemperatur (TEMP) und die Bewegung der Hand in drei Achsen (ACC) misst.

3.2. Transformer statt RNNs

Rekurrente neuronale Netze, insbesondere neuronale Netze mit long short-term memory (LSTM) [45] und rekurrente Netze mit Gattern [46], haben sich bei der Modellierung von Sequenzen und bei Transduktionsproblemen wie der Sprachmodellierung und der maschinellen Übersetzung als State of the Art etabliert. Seitdem wurden zahlreiche Bemühungen unternommen, um die Grenzen von rekurrenten Sprachmodellen und Encoder-Decoder Architekturen voranzutreiben [13].

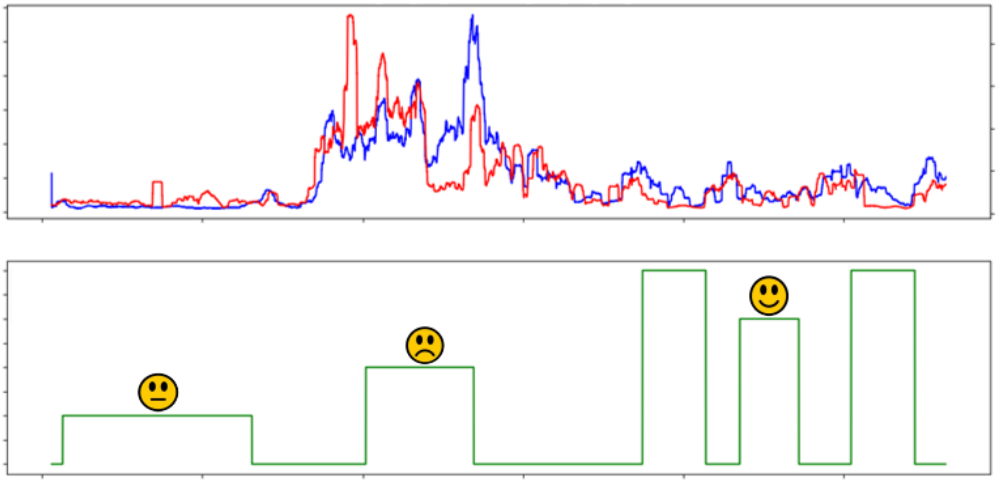


Abbildung 3.1.: Aufbau und Label des WESAD-Datensatzes unterteilt in die Kategorien Baseline, Stress und Amusement [12]

Eine repräsentative Arbeit im Bereich Klassifizierung von Biosignaldaten wurde 2017 von Hannun et al. veröffentlicht. Sie sammelten Einkanal-EKG-Daten von tragbaren Displays und verwendeten ein 34-schichtiges residual-CNN, um Signale zu diagnostizieren. Der Ansatz zeigte eine hohe diagnostische Leistung und übertrifft sogar das durchschnittliche Niveau von Kardiologen in der F1-Bewertung [47].

Wie von Che et al. [48] erkannt wurde, wird hier jedoch das Merkmal, EKG als zeitliches Signal, ignoriert. Ein Transformer-Netzwerk kann zeitliche Merkmale erfassen und sich mit Hilfe eines Attention-Mechanismus auf Kontextvektoren konzentrieren [49].

2017 haben Google Brain Mitarbeiter Vaswani et al. in dem Paper „Attention is All You Need“ [13] diese neue Modellarchitektur vorgestellt, welche auf Rekursion verzichtet. Vielmehr wird ein sogenannter Attention-Mechanismus verwendet, um globale Abhängigkeiten zwischen Eingabe und Ausgabe herzustellen. Der Transformer ermöglicht eine wesentlich stärkere Parallelisierung und konnte den Stand der Technik in der Übersetzungsqualität übertreffen.

Im Vergleich zwischen LSTM und Transformer-Architekturen zum Lösen von automatischer Spracherkennung finden Albert Zeyer et al. [50] heraus, dass die Verwendung einer Transformer-Architektur im Vergleich zu einem ähnlich leistungsfähigen LSTM-Modell weniger Trainingszeit benötigt. Außerdem stellen die Autoren fest, dass das Transformer-Training im Vergleich zum LSTM im Allgemeinen stabiler ist, obwohl es auch zu einem stärkeren Overfitting und damit zu mehr Problemen bei der Generalisierung zu führen scheint. Stellt man allerdings zwei LSTM-Layer dem Transformer-Encoder voran, soll dies ein besseres positional encoding bieten.

3.3. Transformer zur Stresserkennung

Stresserkennung mit Hilfe von Biosignaldaten, wie eben dem WESAD-Datensatz, ist ein aktuelles Forschungsthema. In den Arbeiten von Behinaein et al. [51] und Wand et al. [52] aus den Jahren 2021 bzw. 2022 werden zwei Transformer-Architekturen vorgestellt, mit denen eine Klassifizierung

Tabelle 3.1.: Vergleich der Genauigkeit (%), des F1-Scores (%) und der Standardabweichung (%) - falls angegeben - bei der Klassifikation vom multimodalen Fall, Baseline-, Stress- und Amusement-Situationen, im WESAD-Datensatz unter Berücksichtigung aller Signale.

Referenz	angewandte Methode	Genauigkeit	F1-Score
Garg et al. [53]	Random Forest	67.56	65.73
Schmidt et al. [12]	AdaBoost	79.57 ± 0.93	68.85 ± 0.89
Bobade et al. [54]	ANN	84.32	78.71
Wand et al. [52]	Husformer	85.02 ± 1.91	85.85 ± 2.14
Gil-Martin et al. [11]	CNN	85.03 ± 0.22	85.01 ± 0.22
Bajpaj et al. [55]	k-nearest neighbor	90.00	**90.00**
Behinaein et al. [51]	Transformer	**91.10**	83.30

Tabelle 3.2.: Vergleich der Genauigkeit (%), des F1-Scores (%) und der Standardabweichung (%) - falls angegeben - bei der Klassifikation vom binären Fall, Stress- und kein-Stress-Situationen, im WESAD-Datensatz unter Berücksichtigung aller Signale.

Referenz	angewandte Methode	Genauigkeit	F1-Score
Garg et al. [53]	Random Forest	83.34	84.17
Schmidt et al. [12]	LDA	92.28	90.74
Bobade et al. [54]	ANN	94.24	95.21
Gil-Martin et al. [11]	CNN	**96.62** ± 0.11	**96.63** ± 0.11

von Biosignaldaten möglich ist. Erstere verwendet hierbei ausschließlich EKG-Daten, während der sogenannte *Husformer* aus der zweitgenannten Arbeit multimodale Datenströme aus verschiedensten Sensoren vereinigt. Beide erreichen beim WESAD-Datensatz einen F1-Score zwischen 83 und 86 Prozent. In der Architektur von Behinaein et al. wird eine Kombination aus Convolutional- und Transformer-Architekturen verwendet, wobei die Convolutional-Layer aus lediglich zwei Blöcken besteht. Dies sei nach Aussage der Autoren auf diesem Gebiet eine kleine Anzahl. Die Kernkomponenten vom *Husformer* sind ein cross-modal attention module und das self-attention module. Zum einen werden hierdurch latente Interaktionen zwischen den Modalitäten, zum anderen aber auch die Kontextinformationen im später zusammengeführten Datenstrom gewichtet.

3.4. Übersicht verwandter Ergebnisse zur Stresserkennung in Smartwatch-Daten

Im Folgenden werden die Ergebnisse von einigen Arbeiten, die sich mit der Stresserkennung im WESAD-Datensatz beschäftigt haben, zusammengefasst. Anzumerken ist, dass die meisten Methoden auf den kompletten Datensatz angewendet und somit auch nur Ergebnisse zum multimodalen Fall über alle Signale hinweg ausgeschrieben wurden. Um zunächst den Unterschied zwischen binärer und multimodaler Klassifizierung unter Berücksichtigung aller Signale, aber auch den binären Fall nur unter Berücksichtigung der Signale der tragbaren Smartwatch E4 aufzuzeigen, werden die Ergebnisse in getrennten Tabellen dargestellt.

Tabelle 3.3.: Vergleich der Genauigkeit (%), des F1-Scores (%) und der Standardabweichung (%) - falls angegeben - bei der Klassifikation vom binären Fall, Stress- und kein-Stress-Situationen, im WESAD-Datensatz nur unter Berücksichtigung der Signale von der Smartwatch E4.

Referenz	angewandte Methode	Genauigkeit	F1-Score
Schmidt et al. [12]	LDA	86.88	84.05
Gil-Martin et al. [11]	CNN	**92.70** ± 0.16	**92.55** ± 0.16

Bei dieser Auswahl an Arbeiten erreichen Behinaein et al. [51] mit ihrem *Stresstransformer* bei der Klassifizierung in drei Klassen dabei die höchste Genauigkeit von 91,10 Prozent mit einem F1-Score von 83,30 Prozent. (vgl. Tabelle 3.1) Für die binäre Klassifizierung haben Gil-Martin et al. [11] mit einem CNN-Modell die besten Ergebnisse mit einer Genauigkeit von 96,62 Prozent und einem F1-Score von 96,63 Prozent bzw. 92,70 Prozent und 92,55 Prozent erreicht. (vgl. Tabellen 3.2 und 3.3)

3.5. Differential Privacy

Elektrokardiogrammsignale (EKG) liefern eine Fülle von Informationen über mögliche Herz-Kreislauf-Erkrankungen, von der koronaren Herzkrankheit bis hin zum Risiko eines Herzinfarkts. Während Gesundheitsdienstleister diese Informationen für medizinische und Forschungszwecke speichern und weitergeben, ist der Inhalt dieser Daten, ähnlich wie der Inhalt vieler anderer Arten von Gesundheitsdaten, sehr anfällig für Datenschutzbedenken [56].

Perez et al. [57] identifizieren in ihrer Arbeit aus dem Jahr 2018 verschiedene Datenschutzprobleme im Zusammenhang mit Verbraucher-Wearables wie zum Beispiel unautorisierte Datennutzung, unzureichende Sicherheitsvorkehrungen und fehlende Transparenz bezüglich der Datenverarbeitung.

Um die Privatsphäre zu schützen, wurde unter anderem Differential Privacy als führendes Konzept für statistische Analysen entwickelt. Es ermöglicht die Durchführung komplexer Berechnungen in großen Datenbeständen bei gleichzeitiger Begrenzung der Offenlegung von Informationen über einzelne Datenpunkte [58]. Das Konzept wurde erstmals im Jahr 2008 von Dwork et al. [28] eingeführt.

Hassan et al. [59] sammeln in ihrem Survey zu Differential Privacy in Cyber Physical Systems unter anderem auch wissenschaftliche Arbeiten im Bereich Health and Medical Systems. Beispielsweise entwickelten Zhang et al. [60] den Re-DPoctor-Algorithmus, der Gesundheitsdaten von Wearable-Devices in Echtzeit freigeben kann, während er gleichzeitig die Privatsphäre der betroffenen Patienten durch Differential Privacy schützt. Nguyên et al. [61] stellen eine Methode vor, um Daten von Smart-Geräten-Nutzern unter Verwendung von lokaler Differential Privacy zu erfassen und zu analysieren, um die Privatsphäre der Nutzer zu schützen. Die vorgestellte Methode basiert auf der Aggregation von Rauschen auf den Geräten selbst, anstatt Daten an einen zentralen Server zu senden, und zeigt, dass sie eine effektive Möglichkeit bietet, nützliche Informationen aus Smart-Geräten zu gewinnen, während sie gleichzeitig die Privatsphäre der Nutzer schützt.

In der Praxis werden datenschutzfreundliche Modelle durch eine Wiederholung des Trainings mit einem anderen Optimierer erreicht, wobei jedoch die Modellarchitekturen verwendet werden, die bereits in einer nicht-datenschutzfreundlichen Umgebung gut funktionierten. Um die Genauigkeit der angewandten Methode weiter zu verbessern, schlagen Papernot et al. [62] in ihrem Paper aus dem Jahr 2021 vor, zusätzlich Aktivierungsfunktionen zu nutzen, die explizit für das Training unter Wahrung der Privatsphäre entwickelt wurden. Ein wegweisendes Beispiel hierfür ist der differentiell private stochastische Gradientenabstieg (DP-SGD) von Abadi et al. [63]. Yousefpour et al. veröffentlichen 2019 ihre Differential-Privacy-Implementierung im Rahmen einer PyTorch-Library names *Opacus* welche unter anderem die Anbindung an LSTM-Netzwerken unterstützt [58].

Ungeachtet der Vorteile der Anonymisierung von Daten verringert die Anwendung von Differential Privacy die Genauigkeit der angewandten Methode. Bagdasaryan et al. [64] sammeln in ihrer Arbeit Nachteile, die eine Anwendung von Differential Privacy mit sich bringt. Unter anderem werden unterrepräsentierte Untergruppen unverhältnismäßig stark negativ beeinflusst. Darüber hinaus sinkt die Genauigkeit von DP-Modellen tendenziell stärker in Klassen, die bereits im ursprünglichen, nicht-DP-Modell eine geringere Genauigkeit aufweisen.

4. Methodik

4.1. Problembeschreibung

Die Privatsphäre-erhaltende Stresserkennung in Biosignaldaten in Form von Smartwatch-Health-Daten aus dem WESAD-Datensatz stellt einen vor zwei grundlegende Herausforderungen. Zunächst muss eine ML-Architektur gefunden werden, die eine Klassifizierung in zeitbezogenen Signalen zulässt, wobei möglichst viele Kontextinformationen berücksichtigt werden sollen. Im Anschluss soll das Datenschutzproblem behandelt werden, indem das zuvor gebildete Modell um eine datenschutzfreundliche Lösung erweitert wird. Die sensiblen medizinischen Daten sollen im Endergebnis gemäß den Differential Privacy-Kriterien (vgl. Kapitel 2.3) gegen datenschutzgefährdende Angriffe geschützt sein.

Die Entwicklung einer ML-Lösung zur Stresserkennung untergliedert sich in mehrere Teilaufgaben. Dabei soll zuerst ein funktionsfähiger Prototyp erstellt werden, der die Vorteile der Signalvorverarbeitung aus der Arbeit von Gil-Martin et al. [11] aufgreift und den vollständigen ML-Prozess von der Eingabe der Daten bis zum Training eines Klassifikators abbildet. Das daraus resultierende, nicht-private Modell soll mittels Hyperparameter Tuning weitesgehend verbessert werden. Dafür muss die Auswirkung der einzelnen Parameter auf die Performance des Modells sukzessiv untersucht werden. Da es sich um einen Datensatz mit mehreren Subjekten handelt, empfiehlt es sich, die Leave-One-Subject-Out (LOSO)-Methode zum Training des Modells zu verwenden.

Im nächsten Schritt soll das nicht-private Modell mittels Differential Privacy Privatsphäre-erhaltend werden. Hierfür soll untersucht werden, welcher Trainings-Optimierer sich zur Lösung des Problems eignet und welche Eingabeparameter nötig sind, um bestimmte Differential Privacy-Kriterien zu erfüllen. Diese sollen mit entsprechenden Funktionen berechnet werden, damit letztendlich eine Privatsphäre-erhaltende ML-Architektur entsteht.

Im Anschluss soll die Performance des nicht-Privatsphäre-erhaltenden Falls mit verschiedenen Epsilon-Werten, um eine Beurteilung des Trade-offs zwischen Privatsphäre und Modellleistung geben zu können, der geschützten Modelle verglichen und diskutiert werden.

4.2. Software und Hardware

In diesem Kapitel wird die Software- und Hardwareumgebung detailliert, in der alle Experimente implementiert und durchgeführt wurden. Diese Beschreibung dient dem Nachvollziehen und Reproduzieren der Ergebnisse.

Grundsätzlich wurden zur Implementierung der Gesamtarchitektur ausschließlich Open-Source-Librarys genutzt, um eine hohe Verfügbarkeit zu garantieren. Auch der genutzte WESAD-Datensatz[1] ist öffentlich zugänglich. Python wurde als Programmiersprache verwendet. Dabei entspringt der

[1] Verfügbar unter `https://ubicomp.eti.uni-siegen.de/home/datasets/icmi18/`

Hauptbestandteil der ML-Programmierung der Keras-Library[2]. Keras ist eine in Python geschriebene Deep-Learning-API, die auf der Plattform für maschinelles Lernen TensorFlow[3] aufbaut [65]. Um Differential Privacy nutzen zu können, wurde hier die Tensorflow Privacy Bibliothek[4] genutzt.

Der zugehörige Code wurde mit Hilfe eines Jupyter Notebooks in Google Colab entwickelt und ist in einem GitHub-Repository[5] öffentlich zugänglich. Jupyter ist ein Open-Source-Projekt, das mittels Code in vielen verschiedenen Programmiersprachen arbeiten kann. Verschiedene Sprach-Backends, sogenannte Kernels, kommunizieren mit Jupyter über ein gemeinsames, dokumentiertes Protokoll. Um ein Programm ausführen zu können, benötigt ein User lediglich einen Webbrowser [66].

Die Hardwareressourcen haben einen großen Einfluss auf die Laufzeit und Speichernutzung des Modells. In der Konsequenz bestimmt die genutzte Hardware auch Parametereinstellungen wie die Batch-Size, Epochenzahl oder sogar die ganze Architektur [67]. In dieser Arbeit wird die Hardware von Google Colab[6] vorgegeben. Google Colab ist ein Forschungsprojekt für das Prototyping von Machine-Learning-Modellen auf leistungsstarker Hardware wie GPUs und TPUs. Es bietet eine serverlose Jupyter-Notebook-Umgebung für die interaktive Entwicklung [68]. Alle Experimente wurden mit der Grafikkarte NVIDIA V100 Tensor-GPU mit einem erhöhten Arbeitsspeicher von 25 GB durchgeführt.

4.3. Signalvorverarbeitung

Im Rahmen der vorliegenden Arbeit wurde die Signalvorverarbeitung basierend auf den Erkenntnissen von Gil-Martin et al. [11] aufgebaut. Nachfolgend wird die Wahl der angewandten Methode begründet. Diese wurde so gewählt, dass eine optimale Nutzung des Modells gewährleistet werden konnte.

Die Grundstruktur der Signalvorverarbeitung kann in drei Schritte unterteilt werden: Datenaufbereitung, Segmentierung und Anwendung der Fast-Fourier-Transformation.

Im Rahmen der Datenerfassung wurden die vier Signalarten der *Empatica E4* (vgl. Kapitel 3.1) mit unterschiedlichen Abtastraten erfasst. Um die Daten für die Analyse konsistent zu gestalten, wurden alle Signale innerhalb der Datenaufbereitung auf 64 Hz geupsampelt. Das Upsampling der Daten erfolgte hier durch die Anwendung der Fourier-Methode. Dabei wird das ursprüngliche Signal in den Frequenzbereich transformiert, indem die Fourier-Transformation angewendet wird. Anschließend werden zusätzliche Frequenzpunkte hinzugefügt, um die gewünschte, höhere Abtastrate zu erreichen. Schließlich wird das erweiterte Frequenzspektrum zurück in den Zeitbereich transformiert, um das geupsampelte Signal zu erhalten. Diese Methode ermöglicht es, die Daten aus verschiedenen Sensoren mit unterschiedlichen Abtastraten auf eine einheitliche Abtastrate zu bringen. Durch das Angleichen der Abtastraten wird sichergestellt, dass alle Messungen der verschiedenen Sensoren in gleichen Zeitintervallen vorliegen und für die weiteren Verarbeitungsschritte, wie die Signalvorverarbeitung und die Anwendung der Transformer-Architektur, geeignet sind. Es

[2]Verfügbar unter `https://github.com/keras-team/keras`
[3]Verfügbar unter `https://github.com/tensorflow/tensorflow`
[4]Verfügbar unter `https://github.com/tensorflow/privacy`
[5]Verfügbar unter `https://github.com/BDegenkolb/Privacy-Preserving-Stress-Transformer`
[6]Verfügbar unter `https://colab.research.google.com`

Tabelle 4.1.: Verarbeitungsdetails und Berechung der Subwindow-Länge pro Signal der *Empatica E4* [11].

Signal	**Abtastfrequenz**	**Frequenzbereich**	**Subwindow Länge**	**Anzahl der Inputs für ML-Modell**
Beschleunigung (X, Y, Z)	64 Hz	0 - 30 Hz	7 Sekunden	210
BVP	64 Hz	0 - 7 Hz	30 Sekunden	210
EDA	64 Hz	0 - 7 Hz	30 Sekunden	210
TEMP	64 Hz	0 - 6 Hz	35 Sekunden	210

ist wichtig zu beachten, dass das Upsampling allein keine zusätzlichen Informationen über das Signal liefert. Es dient lediglich dazu, eine einheitliche Abtastrate für die Analyse der verschiedenen Sensordaten zu erreichen, um deren Vergleichbarkeit und Konsistenz zu gewährleisten. Da es durch das künstliche Hinzufügen von Daten zu einer Verschiebung der Labels innerhalb des Datensatzes kommen kann, muss dies im Upsampling berücksichtigt und entsprechend verarbeitet werden.

Im nächsten großen Abschnitt, der Segmentierung, werden die Daten in Fenster (*windows*) und Unterfenster (*subwindows*) unterteilt, um lokale Merkmale innerhalb dieser Fenster zu analysieren. Dabei wird eine Fenstergröße von 60 Sekunden mit einer Verschiebung von 0,25 Sekunden festgelegt. Diese Entscheidung basiert auf früheren Studien [12, 69] und eigener Analyse von Gil-Martin et al. [11], um ein ausgewogenes Verhältnis zwischen der zeitlichen Auflösung und der Genauigkeit der Stresserkennung zu erreichen. Die Länge der Unterfenster hängt dabei vom Signal ab und wird angepasst, um ein durchschnittliches Spektrum mit derselben Anzahl von Frequenzpunkten, hier 210, zu erzeugen. (vgl. Tabelle 4.1)

Wie in Abbildung 4.1, in der die angewandten Signalvorverarbeitungsschitte beispielhaft an einem Signal skizziert sind, zu sehen ist, wird die Fast-Fourier-Transformation (FFT) im Anschluss auf jedes Unterfenster angewendet, um das Spektrum zu erhalten, das entlang aller Unterfenster in einem 60-Sekunden-Fenster gemittelt wird.

4.4. Begründung der Transformer-Architektur

Im Anschluss an die Ausarbeitung der Signalvorbereitung war der nächste Schritt die Wahl eines geeigneten ML-Modells zur Klassifizierung von Stress im WESAD-Datensatz. Entschieden wurde sich schlussendlich für eine Transformer-Architektur. Folgendes Kapitel befasst sich daher mit der Begründung dieser, ihres Ursprungs als auch den notwendigen Schritten zur Anpassung dieser an die vorliegende Problemstellung.

Zur Lösung eines Klassifikationsproblems mit Hilfe eines ML-Modells können viele Architekturen genutzt werden. Die Wahl der Architektur hängt dabei von verschiedenen Faktoren ab, die in der Literatur stark diskutiert werden [70, 71, 72, 73, 74]. Einige wichtige Faktoren, die bei der Auswahl

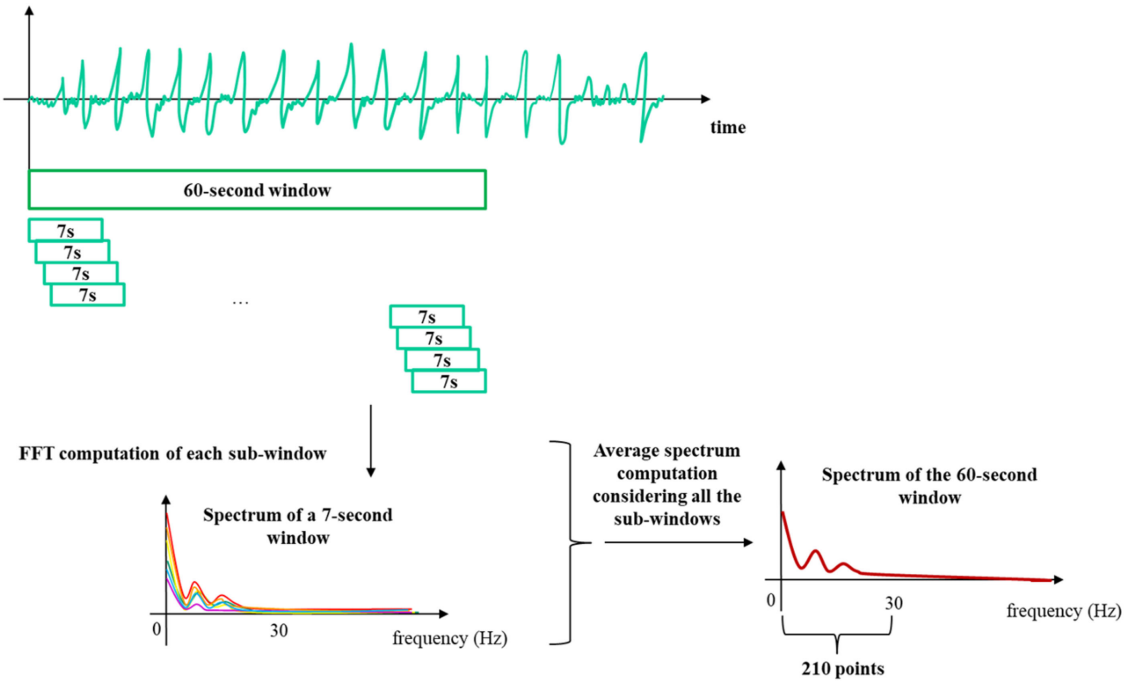


Abbildung 4.1.: Signalvorverarbeitungsschritte am Beispiel eines Signal [11].

des Modells berücksichtigt werden sollten, sind Datensatzgröße, Anzahl der Merkmalsdimensionen, Komplexität des Problems, Interpretier- und Erklärbarkeit, Rechenressourcen als auch Trainings- und Vorhersagezeiten.

Die Größe des zur Verfügung stehenden Datensatzes ist entscheidend für die Wahl des Modells. Einige Modelle wie beispielsweise Deep Learning-Modelle erfordern große Mengen an Daten, um gut zu funktionieren, während andere Modelle wie Entscheidungsbäume oder Support Vector Machines auch bei kleineren Datensätzen gut abschneiden können. Auch die Anzahl der Merkmale (Features) in den Eingabedaten beeinflusst die Wahl des Modells. Bei vielen Merkmalen kann die Verwendung von Modellen wie Random Forests oder Gradient Boosting Machines vorteilhaft sein. Bei hochdimensionalen Daten kann die Anwendung von Dimensionalitätsreduktionstechniken wie PCA (Principal Component Analysis) hilfreich sein, bevor ein Klassifikationsmodell angewendet wird. Die Art des Klassifikationsproblems und die Beziehungen zwischen den merkmalen beeinflussen die Wahl dahingehend, dass bei linearen Beziehungen Modelle wie die lineare Regression oder logistische Regression angemessen sein können. Wohingehend bei komplexeren, nichtlinearen Beziehungen Modelle wie künstliche neuronale Netze oder Ensemble-Methoden wie Random Forests und Gradient Boosting Machines besser geeignet sein können. In einigen Anwendungen ist es wichtig, dass das Modell leicht verständlich und interpretierbar ist. Modelle wie Entscheidungsbäume, lineare Regression und logistische Regression sind in der Regel einfacher zu interpretieren als komplexe Modelle wie Deep Learning-Modelle. Modelle wie künstliche neuronale Netze erfordern eine hohe Rechenleistung und können von der Nutzung von GPUs profitieren. Andere Modelle wie Entscheidungsbäume oder Naive Bayes sind weniger rechenintensiv. Die Zeit, die zum Trainieren des Modells und zum Generieren von Vorhersagen benötigt wird, kann für manche Anwendungen entscheidend sein. Viele Modelle wie künstliche neuronale Netze benötigen eine längere Trainingszeit, während andere wie Entscheidungsbäume oder Naive Bayes schneller trainiert werden können.

Die Transformer-Architektur, die erstmals im Paper *Ättention is All You Need"* von Vaswani et al. [13] vorgestellt wurde, hat sich in vielen Anwendungen, einschließlich Klassifikationsproblemen, als leistungsstark erwiesen. Dabei ist es wichtig zu beachten, dass Transformer-Modelle ursprünglich für NLP-Aufgaben (Natural Language Processing) entwickelt wurden. Für Zeitreihen- oder Sensorbasierte Daten wie im WESAD-Datensatz ist eine Anpassung der ursprünglichen Transformer-Architektur für Zeitreihenklassifikation notwendig. Diese Time-Series-Classification-Transformer-Architektur hat die Besonderheit, keinen Decoder-Teil zu besitzen. (vgl. Kapitel 2.2) Durch die Berücksichtigung der zeitlichen Abhängigkeiten und Strukturen in den Daten ist diese Architektur besser für Probleme wie die Klassifizierung von Stress im WESAD-Datensatz geeignet. Eine beispielhafte Anwendung der Implementierung eines TSCT[7], auf der die Transformer-Architektur in dieser Arbeit letztendlich aufbaut, wurde von Ntakouris [75] zur Verfügung gestellt. Zur Begründung der Anwendung einer TSCT-Architektur werden oben genannte Faktoren aufgegriffen:

1. Datensatzgröße: Time-Series-Classification-Transformer können von großen Datensätzen profitieren. Wenn der WESAD-Datensatz groß genug ist, um die Kapazität des Modells zu nutzen, kann ein Time-Series-Classification-Transformer erfolgreich eingesetzt werden.

2. Merkmalsdimensionen: Der WESAD-Datensatz enthält mehrere Sensormerkmale, die die physiologischen Zustände der Probanden erfassen. Die TSCT-Architektur kann möglicherweise Muster und Zusammenhänge zwischen diesen Merkmalen erkennen, insbesondere wenn sie richtig skaliert und angepasst sind.

3. Komplexität des Problems: Die Klassifizierung von Stress kann ein komplexes Problem sein, das von einer Vielzahl von Faktoren abhängt. Die Flexibilität der TSCT-Architektur ermöglicht es, komplexe Beziehungen in den Daten zu erkennen, die für die Stresserkennung relevant sein können.

4. Interpretierbarkeit und Erklärbarkeit: Die Erklärbarkeit von Time-Series-Classification-Transformer-Modellen kann eine Herausforderung darstellen, insbesondere wenn es darum geht, die Beziehung zwischen physiologischen Merkmalen und Stresszuständen zu erklären. Dies wird allerdings im Rahmen dieser Arbeit nicht weiter beachtet.

5. Rechenressourcen: Die Anwendung der Time-Series-Classification-Transformer-Architektur auf den WESAD-Datensatz erfordert ausreichende Rechenressourcen für das Training und die Inferenz. Wenn solche Ressourcen verfügbar sind, kann ein Time-Series-Classification-Transformer-Modell gute Ergebnisse liefern. Die Nutzung der Ressourcen welche von *Google Colab Pro* zur Verfügung gestellt wurden, sollten zur Lösung des Problems ausreichend gut sein.

6. Trainings- und Vorhersagezeiten: Das Training von Time-Series-Classification-Transformer-Modellen auf dem WESAD-Datensatz kann zeitaufwendig sein. In diesem Fall ist es wichtig, die potenziellen Vorteile in Bezug auf die Modellgenauigkeit gegenüber den Trainings- und Vorhersagezeiten abzuwägen.

[7] Verfügbar unter `https://github.com/keras-team/keras-io/blob/master/examples/timeseries/timeseries_classification_transformer.py`

Zusammenfassend kann die Time-Series-Classification-Transformer-Architektur für die Klassifizierung von Stress im WESAD-Datensatz in Betracht gezogen werden, insbesondere wenn die Datensatzgröße und die Rechenressourcen ausreichend sind und die Komplexität des Problems die Verwendung dieser flexiblen Architektur rechtfertigt.

4.5. Anwendung der Transformer-Architektur

Im folgenden Kapitel werden die Schritte beschrieben und begründet, die nötig waren, um gegebenes Transformer-Modell von Keras [75] an das Problem der Stresserkennung im WESAD-Datensatz anzupassen, anschließend ein zur Stressklassifizierung geeigneter Prototyp präsentiert werden kann.

Die Stresserkennung im WESAD-Datensatz unterscheidet sich vor allem in drei Punkten von dem Klassifizierungsproblem am Ford-Datensatz, an dem die TSCT-Architektur beispielhaft von Ntakouris [75] angewendet wurde. Da es sich um zwei verschiedene Datensätze handelt, muss erstens das *input shape* angepasst werden. *Input shape* bezieht sich auf die Dimensionen oder die Struktur der Eingabedaten, die von einem maschinellen Lern- oder Deep-Learning-Modell erwartet werden. Wie in Kapitel 4.3, der Signalvorverarbeitung, beschrieben, werden pro Signal der Smartwatch 210 Inputs generiert.

Zweitens handelt es sich bei der Stresserkennung um ein binäres und nicht, wie im Fall des Ford-Datensatzes, um ein mutlimodales Klassifizierungsproblem. Hier sollten also sowohl die Output-Dimension auf zwei reduziert, als auch die Aktivierungsfunktion in der Ausgabeschicht von softmax zu sigmoid verändert werden. Die Sigmoid-Aktivierungsfunktion ist für binäre Klassifizierungsprobleme besser geeignet als die Softmax-Aktivierungsfunktion, da sie speziell für binäre Probleme konzipiert ist und weniger rechenintensiv ist [76, 77]. Die Sigmoid-Funktion gibt Werte zwischen 0 und 1 zurück, die als Wahrscheinlichkeiten für die Zugehörigkeit zu einer bestimmten Klasse interpretiert werden können.

Die Softmax-Funktion hingegen ist besser für mehrklassige Klassifizierungsprobleme geeignet. Sie gibt Wahrscheinlichkeiten für jede Klasse zurück, die zusammengefasst eins ergeben. Für binäre Klassifizierungsprobleme ist die Verwendung der Softmax-Funktion übermäßig komplex und weniger effizient, da sie mehr Rechenleistung benötigt, um die Wahrscheinlichkeiten für beide Klassen zu berechnen.

Zwar kann die Softmax-Funktion in der Praxis für binäre Klassifizierungsprobleme verwendet werden und ähnliche Ergebnisse wie die Sigmoid-Funktion liefern, jedoch ist zu beachten, dass sie weniger effizient ist und im Allgemeinen nicht empfohlen wird.

Drittens ist der WESAD-Datensatz in Signale pro Proband aufgeteilt. Häufig wird die Leave-One-Subject-Out-Evaluation im Zusammenhang mit der Stresserkennung und anderen biomedizinischen Anwendungen verwendet [12, 78, 79, 80]. Leave-One-Subject-Out-Evaluation ist eine Art der Kreuzvalidierung, bei der alle Daten eines einzelnen Subjekts (auch als Teilnehmer oder Versuchsperson bezeichnet) als Testdaten verwendet werden, während die Daten der verbleibenden Subjekte zum Trainieren des Modells verwendet werden. Dieser Vorgang wird für jedes Subjekt wiederholt, sodass jedes Subjekt einmal als Testdaten dient. Die Ergebnisse werden anschließend gemittelt, um

die Gesamtleistung des Modells zu bewerten. Drei Gründe sprechen dafür, dass sich eine LOSO-Evaluation für das Training und die Evaluierung eines Klassifikationsmodells zur Stresserkennung im WESAD-Datensatz eignet:

1. Interindividuelle Variabilität: Menschen zeigen unterschiedliche physiologische und bewegungsbezogene Reaktionen auf Stress. Durch die Verwendung der LOSO-Evaluation wird das Modell darauf trainiert, generelle Muster zur Stresserkennung zu lernen, die über verschiedene Individuen hinweg anwendbar sind, anstatt sich auf spezifische Merkmale eines einzelnen Subjekts zu konzentrieren.

2. Generalisierung: Da das Modell auf Daten mehrerer Subjekte trainiert wird und die Leistung auf einem ausgeschlossenen Subjekt getestet wird, kann die Fähigkeit des Modells, auf neue, nicht gesehene Daten zu generalisieren, besser bewertet werden. Dies ist besonders wichtig, wenn das Modell später auf Daten von neuen Personen angewendet werden soll.

3. Vermeidung von Überanpassung: Die Trennung von Trainings- und Testdaten auf Subjektebene reduziert das Risiko von Überanpassung (Overfitting) auf die Trainingsdaten, da das Modell nicht in der Lage ist, lediglich subjektspezifische Informationen zur Erkennung von Stress zu nutzen.

Insgesamt ermöglicht die LOSO-Evaluation eine realistische Einschätzung der Modellleistung, wenn es darum geht, Stress in realen Anwendungsszenarien und bei verschiedenen Individuen zu erkennen. Daher muss ein Rahmen in Form von Schleifen geschaffen werden, damit ein LOSO-Training und -Evaluation möglich ist.

4.6. Hyperparameter-Optimierung

Nachdem ein funktionsfähiger Prototyp im Form eines Transformer-Modells exisitiert, gilt es die Performance der Architektur sukzessiv zu verbessern. Dies wird in vorliegender Arbeit durch Hyperparameter-Optimierung erreicht.

Hyperparameter-Optimierung ist der Prozess, bei dem die optimalen Hyperparameter für ein maschinelles Lernmodell ausgewählt werden. Hyperparameter sind Einstellungen oder Parameter, die vor dem Trainingsprozess festgelegt werden und das Lernverhalten sowie die Leistung eines Modells beeinflussen. Im Gegensatz zu Modellparametern, die während des Trainings automatisch angepasst werden, müssen Hyperparameter manuell oder durch spezielle Optimierungstechniken eingestellt werden. [76]

Folgende Parameter stehen zum Optimieren zur Verfügung:

1. *epochs*: Dieser Parameter gibt die maximale Anzahl von Epochen an, für die das Modell trainiert werden soll. Eine Epoche entspricht einer vollständigen Durchsicht des gesamten Trainingsdatensatzes. Durch das Einstellen der Anzahl der Epochen kann das Training angepasst werden, um Overfitting oder Underfitting zu verhindern.

2. *batch_size*: Die Anzahl der Trainingsbeispiele, die in einer einzigen Batch verarbeitet werden. Nachdem das Modell eine Batch verarbeitet hat, wird ein Gradienten-Update durchgeführt. Durch das Einstellen der Batch-Größe können die Rechenanforderungen und die Konvergenzgeschwindigkeit des Modells beeinflusst werden.

3. *lr*: Die Lernrate ist ein Hyperparameter, der die Größe der Schritte bestimmt, die der Optimierer während des Trainings macht. Eine zu hohe Lernrate kann dazu führen, dass das Modell nicht konvergiert, während eine zu niedrige Lernrate das Training verlangsamen kann.

4. *min_delta_loss*: Dieser Wert gibt die Mindeständerung des Verlusts (Loss) an, die als Verbesserung während des Trainings betrachtet wird. Wird im Early-Stopping-Callback verwendet, um das Training zu stoppen, wenn der Verlust nicht mehr signifikant abnimmt. Durch das Einstellen dieses Werts kann verhindert werden, dass das Training unnötig lange dauert oder zu früh abbricht.

5. *num_transformer_blocks*: Dieser Parameter bestimmt die Anzahl der aufeinanderfolgenden Transformer-Blöcke im Modell. Jeder Block besteht aus einer Multi-Head-Self-Attention-Schicht und einer Feed-Forward-Schicht (vgl. Kapitel 2.2). Durch Erhöhen der Anzahl der Blöcke kann das Modell komplexere Muster in den Daten erkennen. Gleichzeitig kann aber auch das Risiko von Overfitting und erhöhten Rechenanforderungen steigen.

6. *mlp_dropout*: Die Dropout-Rate bestimmt den Anteil der Neuronen, die zufällig deaktiviert (ignoriert) werden, um Overfitting zu verhindern. Diese Rate gilt für die Schichten im Multi-Layer-Perceptron (MLP) Teil des Modells. Durch das Einstellen der Dropout-Rate kann die Regularisierung des Modells beeinflusst werden.

7. *dropout*: Dieser Parameter gibt die Dropout-Rate für die Transformer-Encoder-Schichten im Modell an. Wie beim *mlp_dropout* hilft die Anpassung der Dropout-Rate dabei, das Overfitting zu kontrollieren.

8. *num_heads*: Dieser Wert bestimmt die Anzahl der parallelen Köpfe in der Multi-Head-Self-Attention-Schicht innerhalb eines Transformer-Blocks. Mehrere Köpfe ermöglichen es dem Modell, verschiedene Aspekte der Eingabeinformationen gleichzeitig zu berücksichtigen. Durch das Einstellen der Anzahl der Köpfe kann die Modellkapazität und die Fähigkeit, komplexe Muster zu erkennen, beeinflusst werden.

9. *ff_dim*: Dieser Parameter gibt die Anzahl der Filter in der Feed-Forward-Schicht des Transformer-Blocks an. Die Feed-Forward-Schicht besteht aus zwei 1D-Konvolutionsschichten, die die Informationen innerhalb des Transformer-Blocks verarbeiten. Durch das Anpassen der Anzahl der Filter (*ff_dim*) in diesen Schichten kann die Kapazität und die Fähigkeit des Modells beeinflusst werden, komplexe Muster in den Daten zu erkennen.

10. *head_size*: Dieser Parameter legt die Dimension jedes Aufmerksamkeitskopfes in der Multi-Kopf-Self-Attention-Schicht innerhalb eines Transformer-Blocks fest. Die Größe der Aufmerksamkeitsköpfe hat direkten Einfluss auf die Fähigkeit des Modells, feinere Informationen aus den Eingabedaten zu extrahieren und unterschiedliche Aspekte der Eingabe gleichzeitig zu

berücksichtigen. Durch das Anpassen der *head_size* kann die Kapazität des Modells und seine Fähigkeit, komplexe Muster in den Daten zu identifizieren und zu verarbeiten, gesteuert werden.

Um die Leistung des Modells zu optimieren, wurden die Parameter schrittweise einzeln angepasst. Nach jeder Anpassung wurde der Transformer zehnmal erneut trainiert. Dadurch sollten die statistische Robustheit gewährleistet und die Auswirkungen von zufälligen Faktoren im Trainingsprozess minimiert werden. Während des Trainings können zufällige Elemente wie die Reihenfolge der Trainingsdaten, die Initialisierung der Modellgewichte und die Stochastik des Optimierers die Modellleistung beeinflussen.

Durch mehrfaches Training mit unterschiedlichen Initialisierungen können diese zufälligen Einflüsse gemittelt werden. Dadurch wird sichergestellt, dass die beobachteten Verbesserungen in der Modellleistung auf der Änderungen der Parameter und nicht zufälligen Faktoren im Trainingsprozess beruhen. Die Wahl von zehn Wiederholungen bietet einen angemessenen Kompromiss zwischen der Abschätzung der durchschnittlichen Modellleistung und dem erforderlichen Rechenaufwand für das Training.

Der beste Wert, gemessen an der durchschnittlichen Genauigkeit und dem F1-Score, wurde beibehalten und für die Anpassung des nächsten Parameters verwendet. Um mögliche Korrelationen zwischen den Parametern zu identifizieren oder auszuschließen, wurden zusätzlich einige Durchläufe mit verschiedenen Kombinationen von Parameterwerten durchgeführt.

Neben den erwähnten Parametern wurde auch ein Vergleich zwischen einem Training mit und ohne Early-Stopping durchgeführt. Early-Stopping ist eine Technik, die im Maschinenlernen angewendet wird, um das Problem der Überanpassung (Overfitting) zu vermeiden und die Trainingszeit zu reduzieren [81]. Überanpassung tritt auf, wenn ein Modell so gut auf die Trainingsdaten abgestimmt ist, dass es anfängt, das Rauschen der Daten zu lernen und somit seine Fähigkeit, neue, unbekannte Daten korrekt zu generalisieren, beeinträchtigt.

Early-Stopping funktioniert, indem es den Trainingsprozess überwacht und das Training stoppt, sobald eine bestimmte Metrik, wie zum Beispiel der Loss-Wert, auf den Validierungsdaten keine signifikante Verbesserung mehr zeigt. Um sicherzustellen, dass kleine Schwankungen im Loss-Wert nicht sofort zum Anhalten des Trainings führen, wurde eine sogenannte Geduld (*patience*) eingeführt. Geduld ist die Anzahl der aufeinanderfolgenden Epochen, hier zehn, in denen keine Verbesserung beobachtet wird, bevor das Training gestoppt wird.

Während des Trainings wurde das Modell in regelmäßigen Abständen auf den Validierungsdaten evaluiert. Wenn also der Loss-Wert auf den Validierungsdaten über zehn aufeinanderfolgenden Epochen keine Verbesserung mehr zeigt, die größer als ein vorgegebenes minimales Delta ist (vgl. Parameter *min_delta_loss*), wird das Training abgebrochen.

Durch das frühzeitige Beenden des Trainings kann das Modell eine bessere Generalisierungsfähigkeit bewahren und gleichzeitig die Trainingszeit reduzieren [82].

Der Unterschied zum herkömmlichen Training ohne Early-Stopping wurde anhand der Genauigkeit, dem F1-Score und der Trainingsdauer bewertet.

4.7. Vorbereitung Differential Privacy

Nach erfolgreicher Optimierung des Transformer-Architektur-Prototyps anhand der Hyperparameter erfolgte im nächsten Schritt die Anwendung von Differential Privacy auf diese Architektur. Differential Privacy ist ein wichtiger Ansatz, um die Privatsphäre der Teilnehmer in maschinellem Lernen und Datenanalyse zu schützen, indem Rauschen in den Daten eingeführt wird, um die Rückverfolgbarkeit der Informationen zu den individuellen Teilnehmern zu erschweren (vgl. Kapitel 2.3). Bevor Differential Privacy jedoch auf die Transformer-Architektur angewendet werden kann, müssen entsprechende Vorkehrungen getroffen werden. Diese beinhaltet die Festlegung einer festen Anzahl von Epochen. In diesem Unterkapitel werden die notwendigen Vorbereitungen für die Anwendung von Differential Privacy detailliert beschrieben und begründet.

Um Differential Privacy auf die Transformer-Architektur anwenden zu können, ist es notwendig, eine feste Anzahl von Epochen festzulegen. Dies hat zwei Gründe: Zum einen wird durch die Verwendung der gleichen Anzahl von Epochen wie beim Training ohne Differential Privacy eine bessere Vergleichbarkeit gewährleistet, zum anderen wird der Multiplikator des hinzugefügten Rauschens, der *noise multiplier*, zur Erreichung eines bestimmten Epsilons unter anderem anhand der verwendeten Epochenanzahl berechnet.

In einem LOSO-Training wird das gleiche Modell für alle Teilnehmer einzeln trainiert, wobei jedes Mal ein Teilnehmer als Testdatensatz zurückgelassen und der Rest als Trainingsdaten verwendet wird. Wenn Early-Stopping im LOSO-Training angewendet wird, wird das Training des Modells gestoppt, sobald eine festgelegte Anzahl von Epochen keine signifikante Verbesserung der Leistung auf dem Validierungsdatensatz zeigt. Da die Validierungsleistung von den individuellen Teilnehmerdaten abhängt, kann die Anzahl der Epochen, bei denen Early-Stopping eintritt, für verschiedene Teilnehmer unterschiedlich sein. Dies führt dazu, dass das Training bei unterschiedlichen Epochenanzahlen gestoppt wird und somit die Anzahl der Epochen variieren kann, die für das Training des Modells auf den verschiedenen Teilnehmern benötigt werden. Im Umkehrschluss bedeutet das im schlimmsten Fall, dass bei Anwendung des Early-Stoppings das Training bei 15 unterschiedlichen Subjekten auch bei 15 verschiedenen Epochenanzahlen gestoppt wird.

Um eine Anzahl von Epochen zu finden, die die Leistung des ursprünglichen Modells möglichst nicht beeinträchtigt, wurde bei den durchgeführten Läufen mit Early-Stopping die maximale Anzahl der verwendeten Epochen über alle Subjekte hinweg zwischengespeichert. Wurde beispielsweise beim LOSO-Training von Subjekt *zwei* im gesamten Lauf die höchste Anzahl an Epochen erreicht, wird diese am Ende unter dem Parameter *max epochs* gespeichert. Der höchste Wert, der auf diese Weise erreicht wurde, beträgt 108. Dieser Wert wurde für die folgenden Läufe ohne Early-Stopping festgelegt. In weiteren experimentellen Läufen wurde der Wert nach oben und unten variiert. Durchschnittlich wurden die besten Ergebnisse mit dem Wert 110 erzielt.

Aufgrund der Struktur des LOSO-Trainings werden die Daten eines Subjekts in allen Epochen des Trainings der anderen Subjekte gezeigt. Um die Gesamtzahl der Epochen zu berechnen, in denen die Daten einer Person angezeigt werden, muss die festgelegte Anzahl von Epochen mit der Anzahl aller anderen Subjekte multipliziert werden. In diesem Fall wurden 110 Epochen 14 anderen Subjekten präsentiert, die Gesamtzahl der Epochen beträgt daher 1540.

4.8. Anwendung von Differential Privacy

Um die Privatsphäre der Teilnehmer während des Trainings der Transformer-Architektur zu gewährleisten, wird Differential Privacy angewendet.

Zur Berechnung des *noise multipliers* und des genutzten *deltas*, um ein bestimmtes Epsilon zu erreichen, müssen sowohl die Anzahl der Epochen als auch die Anzahl der Eingabedatenpunkte berücksichtigt werden. Der *noise multiplier* und das *delta* sind zwei wichtige Parameter in der Differential Privacy, die zur Kontrolle des Datenschutzniveaus und der Modellgenauigkeit beitragen [28, 63].

Der *noise multiplier* bestimmt die Menge an Rauschen, welches während des Trainings des Modells zu den Gradienten hinzugefügt wird. Dieses Rauschen hilft dabei, die Privatsphäre der Daten zu schützen, indem es verhindert, dass Informationen über einzelne Datenpunkte aus dem trainierten Modell extrahiert werden können. Ein höherer Wert für den *noise multiplier* führt zu mehr Rauschen und damit zu einer besseren Privatsphäre. Allerdings kann dies auch die Genauigkeit des trainierten Modells beeinträchtigen. Umgekehrt führt ein niedrigerer Wert zu weniger Rauschen und möglicherweise besserer Modellgenauigkeit, bietet jedoch weniger Schutz für die Privatsphäre der Daten.

Das *Delta* ist ein weiterer Parameter in der Differential Privacy, der das Privatsphäreniveau quantifiziert. Delta ist ein Wert, der typischerweise nahe Null liegt, und die Wahrscheinlichkeit dafür angibt, dass das Datenschutzniveau, welches durch ein gewähltes Epsilon gewährleistet ist, verletzt wird. In anderen Worten handelt es sich dabei um die Wahrscheinlichkeit, mit der das Modell einen Verstoß gegen die Privatsphäre zulässt. Ein kleineres Delta bietet eine stärkere Datenschutzgarantie, während ein größeres Delta eine schwächere Garantie bedeutet. In den meisten Fällen sollte das Delta so gewählt werden, dass es kleiner ist als die inverse Anzahl der Datensätze, um ein angemessenes Privatsphäreniveau zu gewährleisten.

Durch die Signalvorverarbeitung werden Eingabesignale von sechs Sensoren der Smartwatch, bestehend aus drei Signalen des Gyrosensors, Sensordaten des EDA-Sensors, des Temperatursensors und des BVP-Sensors, auf jeweils 210 Datenpunkte pro Person reduziert. Durch die Multiplikation von $6 * 210 * 15$ ergibt sich die Gesamtzahl der während des Trainings betrachteten Datenpunkte von 18.900. Das resultierende Delta beträgt gemäß der Inversen der Anzahl der Datensätze 10^{-5}.

Für das Training mit Differential Privacy wurden drei verschiedene Epsilon-Werte gewählt: 0,1, 1 und 10. Die Bedeutung von ε wird in Grundlagenkapitel 2.3 näher beleuchtet. Die Wahl der drei Epsilonwerte basiert auf den Erkenntnissen von Nasr et al. [83] und Carlini et al. [84], wobei ein niedrigeres Epsilon ≤ 1 eine stärkere Privatsphäre gewährleistet, während ein höheres Epsilon eine geringere Privatsphäre bietet, aber möglicherweise bessere Modellleistung ermöglicht. Der Transformer wurde für jedes der drei definierten Epsilons in 30 Runs trainiert. Um die Ergebnisse deterministisch zu gestalten und eine Reproduzierbarkeit zu gewährleisten, wurde ein *seed* von 42 festgelegt. Da einige der mit Differential Privacy trainierten Modelle Nullmodelle sind, die für jeden Datenpunkt die gleiche Klassifizierung abgeben, wurden zur Evaluation nur Ergebnisse mit einer Genauigkeit von über 30 Prozent berücksichtigt. Der jeweilige Anteil der Nullmodelle, der in Abhängigkeit des erreichten Epsilons variiert, wird im Ergebniskapitel 6.3.2 näher erläutert.

5. Implementierung

In diesem Kapitel wird die Implementierung der entwickelten Methode detailliert. Diese wurde mit Hilfe von *Jupyter Notebook* in *Google Colab* durchgeführt.

5.1. Signalvorverarbeitung

Im Codeabschnitt *Signalvorverarbeitung* wurde ein Programmteil entwickelt, welcher physiologische Daten von Wearable-Sensoren analysiert, um Stress und Entspannung zu erkennen. Die Daten stammen aus dem WESAD-Dataset und wurden von E4 Empatica-Geräten erfasst. Der Code besteht aus mehreren Funktionen, die zur Vorverarbeitung, Erstellung von Fenstern und Unterfenstern, Berechnung der Fourier-Transformation und Anpassung der Daten an verschiedene Smartwatch-Betriebssysteme dienen. Ziel ist es, ein Modell zu trainieren, das die Erkennung von Stress und Entspannung ermöglicht, indem es auf den berechneten Merkmale zurückgreift.

Dabei wurde die Vorverarbeitung der Input-Daten für das Transformer-Modell der Implementierung von Gil-Martin et al. [11] entnommen und angepasst. Um den Zugang zum Datensatz festzulegen, wird hier eine Verbindung mit *Google Drive* zum Zielordner, der den WESAD-Datensatz enthält, hergestellt. Anschließend wird eine *Subject*-Klasse definiert, um die Daten für jede Versuchsperson besser zu organisieren. Die Klasse bietet Methoden zum Extrahieren von Signalen der *Empatica E4* und zum Erstellen eines Pandas DataFrames, der die vorverarbeiteten Daten der Versuchsperson enthält.

Im nächsten Codeabschnitt werden die physiologischen Daten in Fenster (*windows*) und Unterfenster (*subwindows*) unterteilt, um die zeitlichen Merkmale der Signale zu erfassen und die Leistung des Modells zu verbessern. Die Berechnung der Fenster- und Unterfenstergrößen läuft wie folgt ab:

1. Fenstergröße (*window size*): Es wird eine Fenstergröße von 60 Sekunden definiert, da diese Dauer bereits ausreicht, um die zeitlichen Merkmale der physiologischen Signale zu erfassen, die mit Stressreaktionen in Verbindung stehen [11].

2. Unterfenstergröße (*subwindow size*): Um die zeitliche Auflösung der Signale weiter zu erhöhen und die Leistung des Modells zu verbessern, wird jedes Fenster in mehrere Unterfenster aufgeteilt. Die Unterfenstergröße wird durch die Abtastrate und die gewünschte Anzahl der Zeitabschnitte (*timesteps*) im Unterfenster bestimmt.

Im Anschluss wird die Fourier-Transformation zur Analysierung der Frequenzspektren der Biosignale auf die Unterfenster angewendet. Die Ergebnisse werden gemittelt, um den Durchschnitt dieser Frequenzspektren für jedes Hauptfenster zu erhalten. Somit werden die eigentlichen Sensorsignale in Abschnitte von jeweils 60 Sekunden auf eine diskrete Anzahl von hier 30 Inputdaten in Form von *timesteps* reduziert.

Die vorher definierten Funktionen werden für alle Eingangssignale der Empatica E4 und alle Versuchspersonen durchgeführt, sodass Trainings- und Testdaten für die Leave-One-Out-Cross-Validation erstellt werden. Der Pseudo-Code ist in Algorithmus 1 dargestellt.

Algorithmus 1 Pseudocode für Signalvorverarbeitung [11]

```
Definiere den Pfad zum Datensatz
Definiere die Subject-Klasse
    Extrahiere Daten von Empatica E4
    Erstelle DataFrame mit vorverarbeiteten Daten
Erstelle Dictionary mit allen Versuchspersonen und zugehörigen DataFrames

Definiere Funktionen zum Erstellen von Fenstern und Unterfenstern
    Erstelle Hauptfenster für jede Minute der Aktivität
    Erstelle Unterfenster für verschiedene Biosignale
    Wende Fourier-Transformation auf Unterfenster an
    Bilde Durchschnitt der Fourier-Transformierten

Erstelle Dictionary mit gemittelten Fenstern für alle Versuchspersonen
Extrahiere gemittelte Fenster und zugehörige Labels für jede Versuchsperson

Definiere Funktion zum Filtern der Daten nach Betriebssystem einer Smartwatch
Gebe Daten zurück, die für die Empatica E4 verfügbar sind
```

5.2. Transformer Modell

Statt eines CNN-Modells wird in dieser Arbeit ein Transformer zur Klassifizierung von zeitbezogenen Signalen verwendet. Hierfür bildet eine Implementierung von Keras [75] die Grundlage der Architektur.

Die *build_model*-Funktion erstellt das gesamte Modell, indem sie die Eingabeform (*input_shape*), die Größe der Aufmerksamkeitsköpfe (*head_size*), die Anzahl der Aufmerksamkeitsköpfe (*num_heads*), die Dimension des Feedforward-Netzwerks (*ff_dim*), die Anzahl der Transformer-Blöcke (*num_transformer_blocks*), die MLP-Einheiten (*mlp_units*), den Dropout-Wert (*dropout*) und den MLP-Dropout-Wert (*mlp_dropout*) annimmt.

Die Eingabedaten werden durch mehrere Transformer-Encoder-Blöcke verarbeitet. Diese sind in der Lage, die hierarchische Struktur der Daten und die Beziehungen zwischen verschiedenen Elementen der Sequenzen zu erkennen. Die Anzahl der Transformer-Blöcke (*num_transformer_blocks*) kann variiert werden, um die Tiefe des Modells anzupassen und somit die Kapazität des Modells zur Erfassung komplexer Abhängigkeiten zu steuern. Der Encoder besteht aus zwei Hauptteilen, dem Normalisierungs- und Aufmerksamkeitsteil sowie dem Feedforward-Teil. Ein Auszug aus dem Modell, welcher einen Encoder-Block beinhaltet, wird in Abbildung 5.1 dargestellt.

Im ersten Teil wird zunächst eine Layer-Normalisierung auf die Eingaben angewendet. Anschließend wird eine Multi-Head-Attention-Schicht eingesetzt, um komplexe Abhängigkeiten in den Daten zu erfassen. Darauf folgt ein Dropout, um Überanpassung zu vermeiden. Schließlich werden die Ergebnisse der Aufmerksamkeitsschicht und die ursprünglichen Eingaben addiert, um das Residual

(res) zu bilden. Das Residual ist ein Konzept, das in tiefen neuronalen Netzwerken, insbesondere in Residual-Netzwerken (*ResNets*) und Transformer-Modellen, verwendet wird. Es handelt sich dabei um eine Technik, bei der die Eingabe einer Schicht direkt zur Ausgabe derselben Schicht oder einer späteren Schicht addiert wird. Dies geschieht, um das Gradientenverschwinden-Problem bei tiefen Netzwerken zu mildern und die Konvergenz während des Trainings zu verbessern [85].

Im Feedforward-Teil wird eine weitere Layer-Normalisierung auf das Residual angewendet, gefolgt von einer eindimensionalen Faltungsschicht (*Conv1D*) mit ReLU-Aktivierung. Durch Dropout und eine weitere eindimensionalen Faltungsschicht wird die Ausgabedimensionen auf die ursprünglichen Eingabedimensionen zurückgeführt. Die resultierende Ausgabe wird zum Residual addiert und als Ergebnis des Encoders zurückgegeben.

Nach der Verarbeitung durch alle Transformer-Blöcke wird eine Global Average Pooling-Schicht auf die Ausgabe angewendet, um die Feature-Dimension zu reduzieren. Anschließend wird ein Multilayer Perceptron (MLP) mit mehreren Dense-Schichten und Dropout-Schichten erstellt, um die finale Klassifikation durchzuführen.

Das MLP dient zur Klassifikation der Merkmale, die von den Transformer-Blöcken extrahiert werden. Durch Variation der Anzahl und Größe der MLP-Einheiten (*mlp_units*) kann das Modell an spezifische Anforderungen und Ressourcenbeschränkungen angepasst werden.

Die Ausgabeschicht des Modells beinhaltet ein Dense-Layer mit der Anzahl der Ausgabeklassen (*num_output_class*) und einer Sigmoid-Aktivierungsfunktion. Letztere findet vor allem im Rahmen der binären Klassifizierung Gebrauch.

Die Funktion gibt das erstellte Modell als Keras-Modell zurück.

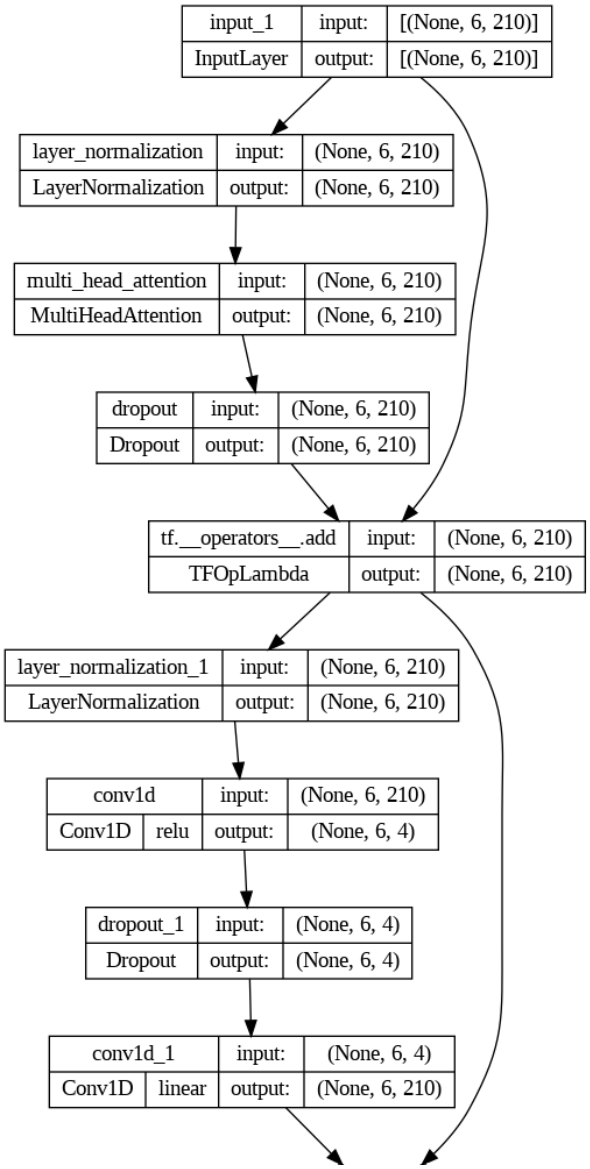


Abbildung 5.1.: Encoder-Block mit Input-Layer. Die Anzahl der hintereinander geschalteten Encoder-Blöcke ist variabel und sollte an das zu lösende Problem angepasst werden.

5.3. Modell-Training

Im Folgenden wird der Code für das Training des zuvor erstellten Transformer-Modells vorgestellt. Dieser enthält den Hauptteil des Trainingsprozesses, einschließlich der Aufteilung der Daten in Trainings- und Testsets, der Verwendung von Differentially Private Optimierern, der Anwendung von Early-Stopping und der Speicherung von Modell-Checkpoints.

Der Abschnitt beginnt mit der Initialisierung von Variablen und dem Setzen des Seeds für TensorFlow. Dieser wird auf 42 gesetzt, um reproduzierbare Ergebnisse zu gewährleisten. Um die Validierung des Modells zu garantieren, wird eine Leave-One-Subject-Out (LOSO)-Methode angewendet, bei der Trainings- und Testsets erstellt werden, indem jeweils ein Subjekt aus dem Datensatz ausgeschlossen wird.

Nachdem die Trainings- und Testdaten erstellt wurden, werden die Daten und Labels in Arrays und die Zielvariablen (Labels) kategorisch umgewandelt. Danach wird das Transformer-Modell erstellt und kompiliert, wobei der Optimierer ausgewählt wird. Hierbei stehen zwei Optimierungsmöglichkeiten zur Verfügung, der normale Adam-Optimierer oder ein differentiell privater Optimierer.

Der Code fügt auch verschiedene Metriken hinzu, um die Modellleistung zu bewerten, darunter Genauigkeit, Precision, Recall und F1-Score. Um die Modellqualität während des Trainings zu überwachen, werden ModelCheckpoint und EarlyStopping Callbacks erstellt. Diese Callbacks helfen dabei, das beste Modell zu speichern und das Training frühzeitig zu stoppen, wenn keine Verbesserungen mehr erzielt werden.

Schließlich wird das Modell mit den Trainingsdaten, den Validierungsdaten, der Anzahl der Epochen, der Batch-Size, der Klassen-Gewichtung und den Callbacks trainiert. Die tatsächliche Anzahl der durchgeführten Epochen wird in einem Dictionary gespeichert, das für jeden Testlauf aktualisiert wird. Dieser gesamte Prozess wird entsprechend der LOSO-Methode für jedes Subjekt im WESAD-Datensatz wiederholt, um ein umfassendes Training und eine Validierung des Modells zu gewährleisten. In Algorithmus 2 wird das Modell-Training in Form von Pseudocode zusammengefasst.

Algorithmus 2 Pseudocode für das Model Training entsprechend der LOSO-Methode

```
Initialisiere Variablen und setze den Seed für TensorFlow
for jedes Betriebssystem in SMARTWATCH_OS do
    Filtere Signale für das aktuelle Betriebssystem
    for jeden Probanden in der Gruppe von Probanden do
        Teile Daten in Trainings- und Testsets, aktueller Proband wird ausgeschlossen (LOSO)
        Berechne das Klassenverhältnis für das Trainingsset
        Konvertiere Trainings- und Testdaten in Arrays und codiere kategorisch Labels
        Baue und kompiliere das Transformer-Modell, wähle Optimierer und füge Metriken hinzu
        Erstelle ModelCheckpoint und EarlyStopping Callbacks
        Trainiere das Modell
        Aktualisiere das Dictionary der durchgeführten Epochen für jeden Testlauf
    end for
end for
```

5.4. Evaluation

In diesem Abschnitt der Implementierung wird die Evaluierung des Modells auf den bisher ungesehenen Testdatensätzen für das E4-Smartwatch-Betriebssystem durchgeführt. Zuerst werden zwei separate Listen erstellt, um die Genauigkeit (Accuracy) und den F1-Score für das E4-Betriebssystem zu sammeln.

Die Iteration erfolgt über die *subject_ids*. Für jeden Teilnehmenden werden die zugehörigen Testdaten (*X_test, y_test*) extrahiert und in NumPy-Arrays konvertiert. Die Zielvariablen (*y_test*) werden anschließend in kategorische Werte umgewandelt.

Der Pfad zum gespeicherten Modell wird generiert, indem der Betriebssystemname (E4), die Anzahl der Epochen und die Teilnehmenden-Id verwendet werden. Falls das Modell noch nicht geladen wurde, wird es vom angegebenen Pfad geladen.

Anschließend wird das Modell auf den Testdaten evaluiert, um die Genauigkeit (Accuracy), Präzision (Precision) und Sensitivität (Recall) zu berechnen. Der F1-Score wird ermittelt, sofern Präzision und Sensitivität nicht gleich Null sind; andernfalls wird der F1-Score auf Null gesetzt.

Alle errechneten Metriken werden in den entsprechenden Listen gespeichert. Nach Abschluss der Iteration werden die Durchschnittswerte für Genauigkeit, Präzision, Sensitivität und F1-Score für das E4-Betriebssystem berechnet und präsentiert.

5.5. Differential Privacy

In diesem Kapitel wird die Implementierung von Differential Privacy in einem Transformer-Modell zur Stresserkennung beschrieben. Differential Privacy schützt die Privatsphäre der Nutzer in maschinellen Lernmodellen, indem es den Zugriff auf individuelle Datenpunkte einschränkt und statistische Informationen über die Daten preisgibt.

Für die Implementierung von Differential Privacy wird der *VectorizedDPKerasAdamOptimizer* aus der TensorFlow Privacy-Bibliothek verwendet, der Parameter wie L2-Norm-Clip, Rauschenmultiplikator und Anzahl der Mikrobatches berücksichtigt. Die Funktionen *compute_noise()* und *compute_delta()* werden verwendet, um den benötigten Rauschenmultiplikator und Delta-Wert für den Trainingsprozess zu berechnen. Der Wert von Delta basiert auf der Anzahl der Trainingsdatenpunkte (n). Delta sollte eine Größenordnung niedriger sein als der Kehrwert der Trainingsdatensatzgröße ($1/n$).

Nach dem Training wird die Differential Privacy des Modells analysiert, indem die Funktion *compute_dp_sgd_privacy()* aus der TensorFlow Privacy-Bibliothek verwendet wird. Dabei werden verschiedene Epsilon-Werte wie 0,1, 1 und 10 untersucht, um den Einfluss des Rauschenmultiplikators auf die Privatsphäre zu bewerten.

Schließlich wird das trainierte Modell auf Testdaten evaluiert, um seine Leistung bei der Stresserkennung zu beurteilen.

6. Evaluation und Ergebnisse

In diesem Ergebniskapitel werden die Evaluationsmetriken und die Leistung der entwickelten Transformer-Architektur sowohl ohne als auch mit Differential Privacy untersucht. Das Kapitel ist so aufgebaut, dass in jedem seiner Unterkapitel ein Aspekt der Evaluation betrachtet wird.

Zunächst werden in Kapitel 6.1 die wichtigsten Evaluationsmetriken vorgestellt, die zur Beurteilung der Modellleistung im nicht-privaten und privaten Fall (mit Differential Privacy) verwendet werden. Anschließend werden in den Kapiteln über die Evaluation der Transformer-Architektur ohne und mit Differential Privacy die Ergebnisse der Hyperparameter-Optimierung, der Trainingsdauer und der Modellperformance im Detail untersucht. Dabei werden sowohl die Leistungsfähigkeit der Transformer-Architektur in ihrer ursprünglichen Konfiguration als auch die Verbesserungen durch gezielte Optimierung hervorgehoben.

Im Kapitel über die Anwendung von Differential Privacy auf die Transformer-Architektur werden die Auswirkungen der Privatsphäre auf die Modellleistung analysiert und mit den Ergebnissen des nicht-privaten Falls verglichen. Dabei wird auch auf die Rolle von Early-Stopping und die Konvergenz der Modelle eingegangen. Insgesamt soll das Kapitel ein umfassendes Verständnis der Leistung der entwickelten Transformer-Architektur unter verschiedenen Bedingungen und Privatsphäre-Anforderungen vermitteln.

6.1. Metriken

In diesem Unterkapitel werden die wichtigsten Evaluationsmetriken erläutert, die für die Bewertung der Leistung des entwickelten Modells sowohl im nicht-privaten Fall (ohne Differential Privacy) als auch im privaten Fall (mit Differential Privacy) eingesetzt werden.

Im nicht-privaten Fall werden verschiedene Evaluationsmetriken verwendet, um unterschiedliche Aspekte der Modellleistung zu bewerten. Die Genauigkeit (*Accuracy*) gibt das Verhältnis der korrekt klassifizierten Datenpunkte zur Gesamtzahl der Datenpunkte an und ist eine grundlegende Metrik zur Beurteilung der Leistung eines Modells [77]. Jedoch kann die Genauigkeit allein ein unvollständiges Bild der Modellleistung liefern, insbesondere wenn es um ungleich verteilte Klassen geht. In solchen Fällen sind Precision und Recall hilfreiche Metriken, um weitere Aspekte der Modellleistung zu bewerten [86].

Precision misst das Verhältnis der korrekt positiv klassifizierten Datenpunkte zur Gesamtzahl der als positiv klassifizierten Datenpunkte [77]. Eine hohe Precision bedeutet, dass das Modell präzise ist und wenige Falsch-Positive aufweist. *Recall* hingegen gibt das Verhältnis der korrekt positiv klassifizierten Datenpunkte zur Gesamtzahl der tatsächlich positiven Datenpunkte an. Ein hoher Recall-Wert gibt an, dass ein Modell die meisten positiven Fälle erfasst und wenige Falsch-Negative aufweist. Die Kombination von Precision und Recall ermöglicht es, ein besseres Verständnis der Modellleistung in Bezug auf unterschiedliche Aspekte der Klassifizierung zu erhalten, insbesondere wenn die Kosten für Falsch-Positive und Falsch-Negative unterschiedlich sind [86].

Um eine zusammenfassende Metrik zu erhalten, die sowohl Precision als auch Recall berücksichtigt, wird der F1-Score verwendet, der das harmonische Mittel von Precision und Recall darstellt [77].

Im privaten Fall, also bei der Anwendung von Differential Privacy, wird vor allem die Metrik Epsilon verwendet, um die Ergebnisse miteinander vergleichen zu können. Epsilon gibt das Maß für den Schutz der Privatsphäre im Zusammenhang mit Differential Privacy an [87]. Ein kleines Epsilon bedeutet eine hohe Privatsphäre, während ein größeres Epsilon auf eine geringe Privatsphäre hinweist. $\varepsilon = \infty$ entspricht dem nicht-privaten Fall, während $\varepsilon \leq 1$ als Kriterium für die Privatsphäre angesehen wird [30]. Die Epsilon-Metrik ermöglicht die Beurteilung des Trade-offs zwischen Privatsphäre und Modellleistung bei verschiedenen Differential-Privacy-Ansätzen.

Die Evaluationsmetriken für den nicht-privaten und den privaten Fall sind entscheidend für die Bewertung und den Vergleich der Leistung von Modellen zur Stresserkennung, insbesondere in Bezug auf Genauigkeit, Präzision und Effizienz. Durch die sorgfältige Anwendung dieser Metriken kann eine fundierte Entscheidung über die beste Vorgehensweise für das Training und die Implementierung von Modellen in diesem Bereich getroffen werden. Es ist zu beachten, dass die Wahl der Metriken von der jeweiligen Anwendung und den spezifischen Anforderungen abhängt, die an die Privatsphäre und die Modellleistung gestellt werden. Insgesamt dienen diese Evaluationsmetriken dazu, ein umfassendes Bild der Leistungsfähigkeit und der Privatsphäreeigenschaften des entwickelten Modells für die Stresserkennung zu erhalten, was für eine effektive und verantwortungsbewusste Anwendung von Data Science im Gesundheitsbereich unerlässlich ist.

6.2. Evaluation der Transformer-Architektur ohne Differential Privacy

In folgenden Abschnitten werden die Ergebnisse der durch Hyperparameter-Optimierung verbesserten Modellperformance dargestellt. Vor allem haben dabei die Anzahl der Transformer-Blöcke, die Anzahl der Heads in den Mulit-Head-Attention-Schichten, der MLP-Dropout und der Dropout-Wert maßgeblichen Einfluss auf die Genauigkeit und den F1-Score genommen.

Zu Beginn des Optimierungsprozesses lag die Genauigkeit des Modells ohne Hyperparameter-Optimierung bei etwa 87 Prozent und der F1-Score bei 85 Prozent. Diese Werte verdeutlichen bereits die beachtliche Leistungsfähigkeit der Transformer-Architektur in ihrer ursprünglichen Konfiguration. Durch die gezielte Optimierung der Hyperparameter konnte die Modellperformance jedoch weiter gesteigert werden.

Nach Durchführung der Hyperparameter-Optimierung konnte eine maximale Genauigkeit von 92,13 Prozent und ein F1-Score von 92,18 Prozent erreicht werden. Diese Ergebnisse unterstreichen das Potenzial der Hyperparameter-Optimierung, die Leistung der Transformer-Architektur bei der Stresserkennung noch weiter zu optimieren.

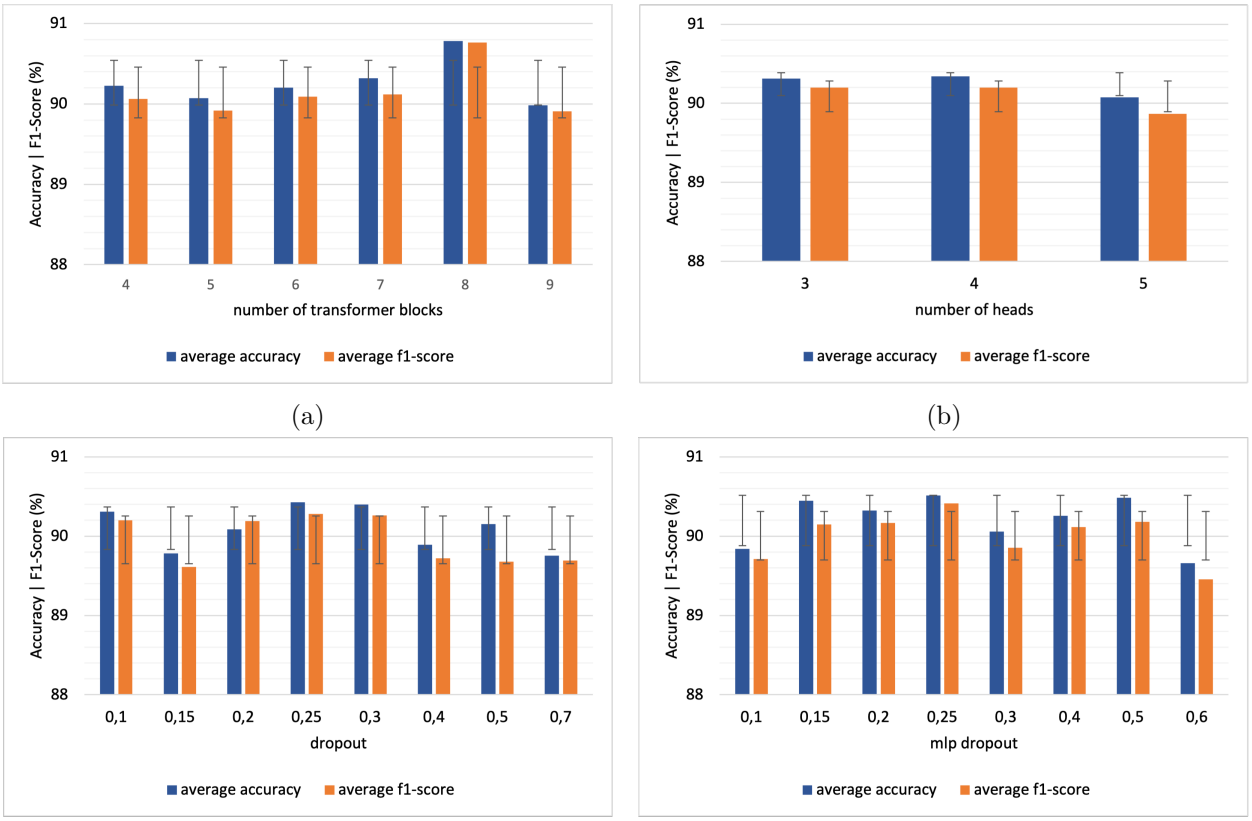


Abbildung 6.1.: Durchschnittliche Modellperformance in Abhängigkeit von (a) der Anzahl der eingesetzten Transformer-Blöcke; (b) der Anzahl der eingesetzten Heads innerhalb der Transformer-Blöcke; (c) dem Dropout-Wert; (d) dem Mlp-Dropout-Wert.

6.2.1. Hyperparameter-Optimierung

Im Rahmen dieses Unterkapitels wird die verbesserte Modellperformance im Detail analysiert, um einen umfassenden Einblick in die Leistungsfähigkeit der optimierten Transformer-Architektur im nicht-privaten Fall zu geben.

Abbildung 6.4 zeigt, wie sich der Wert von vier ausgewählten Hyperparametern, welche den größten Einfluss auf das Endergnis haben, auf die Durchschnittliche Modellperformance auswirken. Dabei wird in Abbildung 6.1a zunächst nur der Einfluss der Anzahl der eingesetzten Transformerblöcke betrachtet. Die besten durchschnittlichen Ergebnisse konnten mit acht Transformer-Blöcken erreicht werden. Dabei wird das Modell mit steigender Anzahl tendenziell besser, verschlechtert sich allerdings nach acht Transformer-Blöcken wieder. Bei acht Transformer-Blöcken erreicht das Modell eine durchschnittliche Genauigkeit von 90,78 Prozent mit einer Standardabweichung (σ) von 0,84 Prozent und einem F1-Score von 90,76 Prozent mit einem σ von 0,85 Prozent.

Abbildung 6.1b zeigt die Auswirkung der Anzahl der Heads pro Transformer-Block. Die Änderung dieser hat nur einen relativ geringen Einfluss auf die durchschnittliche Genauigkeit und F1-Score, die bei drei bis fünf Köpfen zwischen 90,08 Prozent beziehungsweise 89,87 Prozent und 90,34 Prozent beziehungsweise 90,2 Prozent liegen. Die Anzahl der Köpfe ist allerdings in der Trainingsdauer spürbar. Diese wird im folgenden Abschnitt 6.2.2 analysiert.

6.2.2. Trainingsdauer

In Abbildungen 6.1c und 6.1d wird die Modellperformance in Abhängigkeit von dem Hyperparameter Dropout und Mlp-Dropout dargestellt. In beiden Fällen wird beim Wert von 0,25 das beste Ergebnis mit einer Genauigkeit von über 90,4 Prozent und einem σ von 0,77 Prozent beziehungsweise 0,83 Prozent und einem F1-Score von über 90,28 Prozent mit einem σ von 0,81 Prozent beziehungsweise 0,84 Prozent erreicht.

Mit der Einstellung der jeweils besten Werte für die einzelnen Parametern konnte ein Ergebnis mit einer durchschnittlichen Genauigkeit von 91,89 Prozent ($\sigma = 0,13\%$) und einem durchschnittlichen F1-Score von 91,61 Prozent ($\sigma = 0,31\%$) erreicht werden. Der Durchlauf mit der besten Performance erreichte dabei eine Genauigkeit von 92,13 Prozent und einen F1-Score von 92,18 Prozent.

Die Leistung eines Modells kann nicht nur anhand von Metriken wie Genauigkeit und F1-Score bewertet werden, sondern auch anhand der Trainingsdauer. In diesem Unterkapitel wird die Trainingsdauer als zusätzliche Bewertungsgrundlage untersucht, um ein umfassenderes Verständnis der Modellperformance zu erlangen. Dabei spielen neben der Anzahl der Epochen insbesondere die Anzahl der eingesetzten Transformer-Blöcke, die Anzahl der Köpfe innerhalb der Transformer-Blöcke und die Größe der Köpfe eine entscheidende Rolle.

Die Trainingsdauer ist ein wichtiger Faktor, da sie Auswirkungen auf die praktische Anwendbarkeit und Effizienz des Modells hat. Ein Modell, das lange Trainingszeiten benötigt, ist möglicherweise weniger geeignet für Anwendungsfälle, bei denen eine schnelle Anpassung oder Aktualisierung des Modells erforderlich ist.

Abbildung 6.2 zeigt, wie sich die Trainingsdauer in Abhängigkeit von der Anzahl der eingesetzten Transformer-Blöcke, der Anzahl der Heads innerhalb der Transformer-Blöcke und der Größe der Heads verändert. Es ist zu erkennen, dass die Trainingsdauer mit steigender Anzahl aller drei Parameter wächst. Dabei brauchte das schnellste Modell mit nur zwei Transformer-Blöcken knapp sieben Minuten Trainingszeit, während das langsamste Modell mit zehn Transformer-Blöcken fast 26 Minuten zum konvergieren benötigte.

6.3. Evaluation der Transformer-Architektur mit Differential Privacy

Nachdem die Optimierung der Transformer-Architektur im nicht-privaten Fall behandelt wurde und die Ergebnisse evaluiert wurden, soll nun die Anwendung von Differential Privacy auf das optimierte Modell untersucht werden. In diesem Kapitel werden die Ergebnisse der Vorbereitung für die Implementierung von Differential Privacy sowie die Auswirkungen von Differential Privacy auf die Modellperformance im Vergleich zum nicht-privaten Fall präsentiert.

Dieses Kapitel umfasst zwei Unterkapitel, die verschiedene Aspekte der Evaluation behandeln. Im ersten Unterkapitel werden die Ergebnisse der Vorbereitungen für die Anwendung von Differential Privacy vorgestellt. Dazu gehört die Identifizierung einer geeigneten Anzahl von Epochen, welche

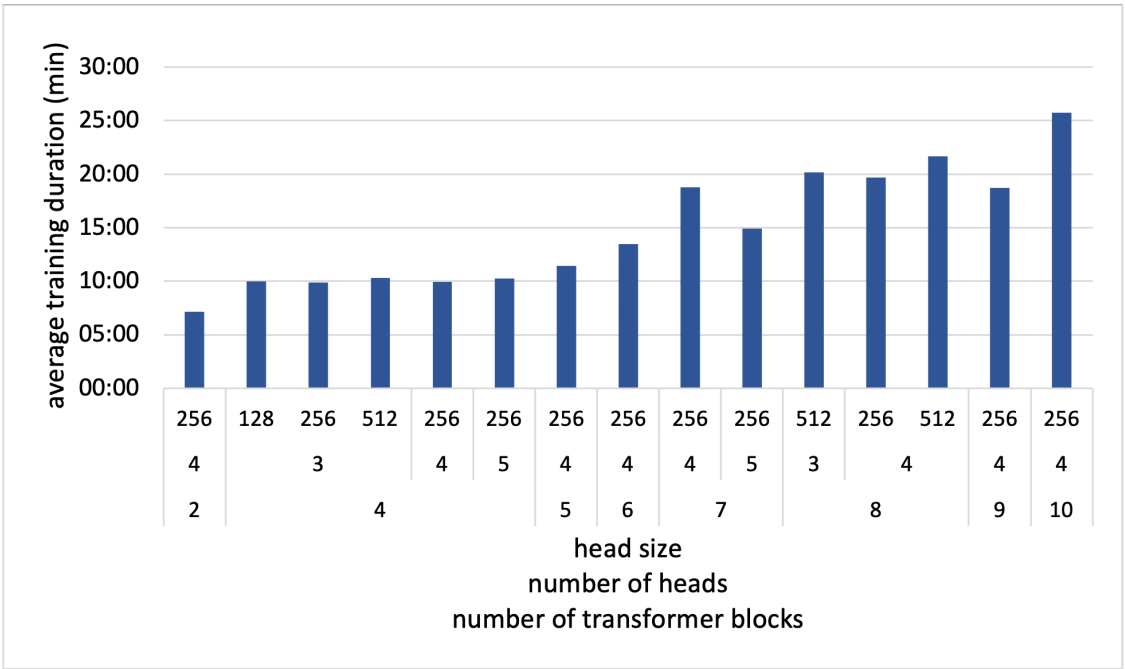


Abbildung 6.2.: Durchschnittliche Trainingsdauer in Abhängigkeit von der Anzahl der eingesetzten Transformer-Blöcke, der Anzahl der eingesetzten Heads innerhalb der Transformer-Blöcke und der Größe der Heads.

die Leistung des ursprünglichen Modells möglichst nicht beeinträchtigt, um unter anderem einen fairen Vergleich mit dem nicht-privaten Fall zu ermöglichen.

Im zweiten Unterkapitel werden die Ergebnisse der Modellperformance nach der Anwendung von Differential Privacy auf das optimierte Modell dargelegt. Hierbei werden die Auswirkungen von Differential Privacy auf die Genauigkeit und den F1-Score analysiert und mit den Ergebnissen des nicht-privaten Falls verglichen. Diese Analyse ermöglicht es, ein umfassendes Verständnis der Auswirkungen von Differential Privacy auf die Modellperformance zu gewinnen und herauszufinden, inwieweit Differential Privacy das Modell beeinflusst.

6.3.1. Modellperformance ohne Early-Stopping

In diesem Unterkapitel der Ergebnisse wird die Modellperformance ohne Early-Stopping im Kontext der Anwendung von Differential Privacy untersucht. Da Early-Stopping für die Implementierung von Differential Privacy deaktiviert werden musste, um eine konstante Anzahl von Epochen für alle Subjekte im LOSO-Training zu gewährleisten, ist es wichtig, die Auswirkungen dieser Änderung auf die Modellperformance zu analysieren.

Um die verschiedenen Aspekte der Modellperformance ohne Early-Stopping zu beleuchten, werden in diesem Kapitel drei Abbildungen hierzu präsentiert. Abbildung 6.3 zeigt die Genauigkeit und den F1-Score in Prozent in Abhängigkeit von der Anzahl der festgelegten Epochen. Aus dieser Darstellung geht hervor, dass die beste Performance bei einer Epochenzahl von 110 erreicht wird.

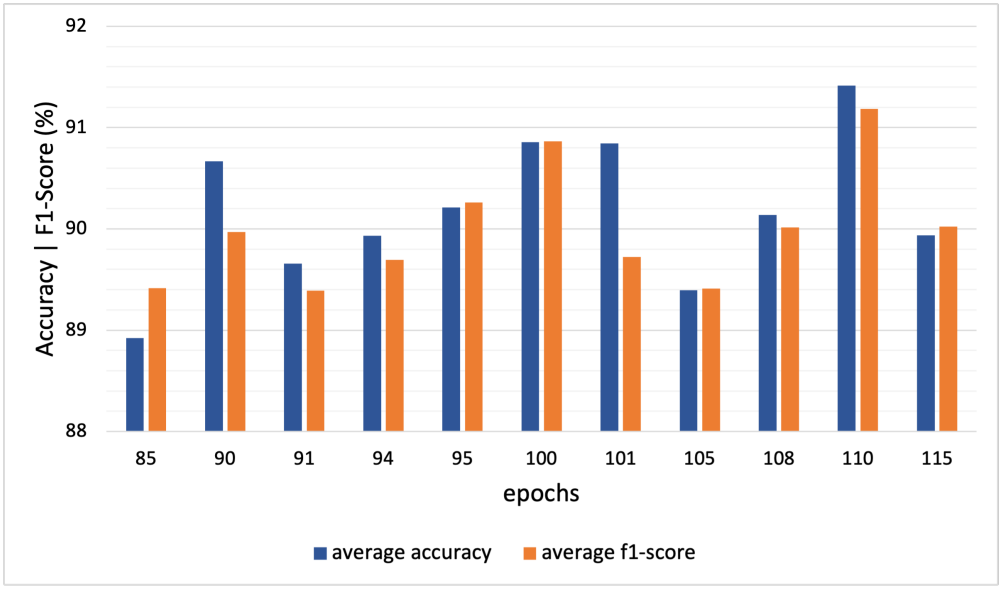
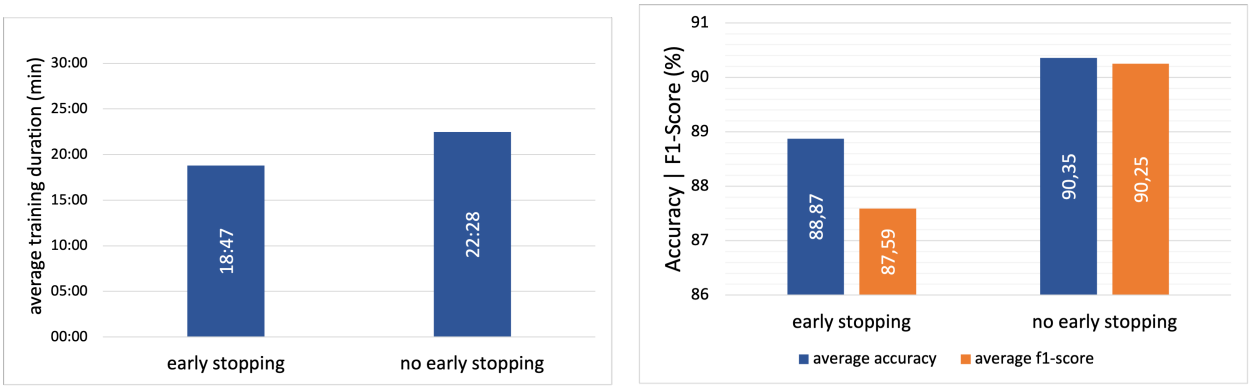


Abbildung 6.3.: Durchschnittliche Performance des Modells in Anhängigkeit von einer festgelegten Anzahl von Epochen ohne Early-Stopping.

In Abbildungen 6.4a und 6.4b wird das Modell mit und ohne Early-Stopping hinsichtlich Trainingsdauer und Performance verglichen, wobei die Anzahl der Transformer-Blöcke und die Anzahl der Heads in den Blöcken konstant gehalten wurden. Es zeigt sich, dass die durchschnittliche Trainingsdauer mit Early-Stopping bei 18 Minuten und 47 Sekunden lag, während sie ohne Early-Stopping bei 22 Minuten und 28 Sekunden lag – ein Unterschied von etwa 20 Prozent!



(a) (b)

Abbildung 6.4.: Vergleich zwischen der Modellergebnisse mit und ohne Early-Stopping bezüglich (a) der benötigten Trainingsdauer, (b) der durchschnittlichen Genauigkeit und zugehörigem F1-Score.

In Hinblick auf die Performance wurde deutlich, dass das Modell mit Early-Stopping eine durchschnittliche Genauigkeit von 88,87 Prozent und einen durchschnittlichen F1-Score von 87,59 Prozent erreichte. Im Gegensatz dazu erzielte das Modell ohne Early-Stopping eine höhere durchschnittliche Genauigkeit von 90,35 Prozent sowie einen durchschnittlichen F1-Score von 90,25 Prozent.

Die Ergebnisse verdeutlichen, dass die Deaktivierung von Early-Stopping für die Anwendung von Differential Privacy zwar zu einer längeren Trainingsdauer führt, aber gleichzeitig eine Verbesserung der Modellperformance erzielt werden kann.

6.3.2. Modellperformance unter Anwendung von Differential Privacy

Im Folgenden werden die Auswirkungen der Anwendung von Differential Privacy auf die Genauigkeit und den F1-Score analysiert und mit den Ergebnissen des nicht-privaten Falls verglichen. Dabei wird das Modell für drei verschiedene Epsilons (0,1, 1 und 10) sowie für den Fall ohne Differential Privacy ($\varepsilon = \infty$) betrachtet. Die Veranschaulichung dieser Vergleiche wird in Abbildung 6.5 dargestellt, die die Performance der verschiedenen Epsilons gegenüberstellt.

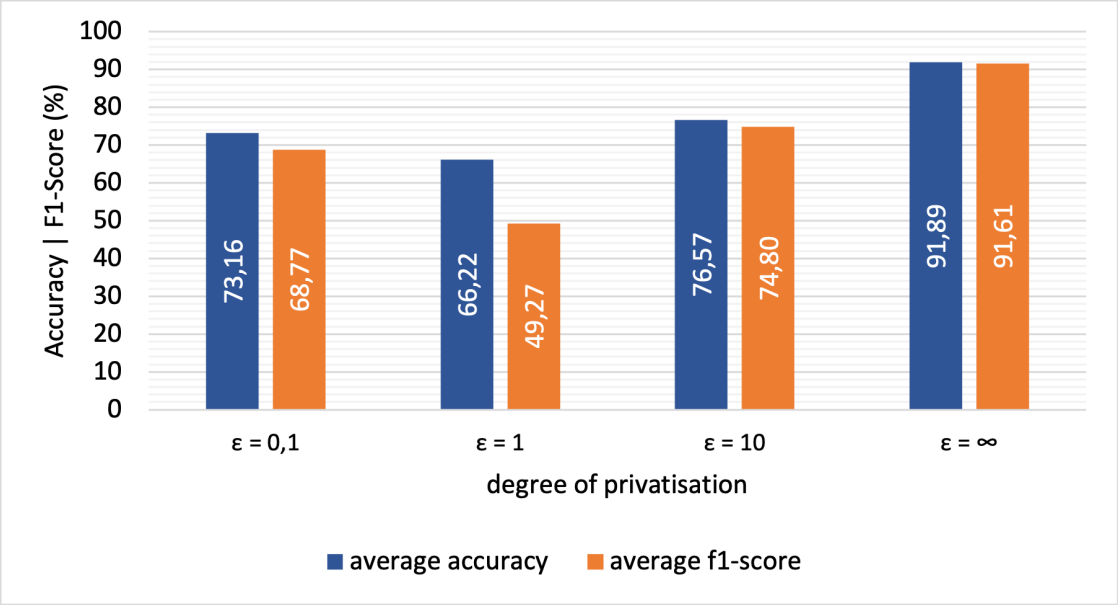


Abbildung 6.5.: Durchschnittliche Performance der Modelle mit Differential Privacy im Vergleich zum Modell ohne Differential Privacy ($\varepsilon = \infty$)

.

Das Modell mit einem niedrigen Privatisierungsgrad $\varepsilon = 10$ erreichte dabei unter den Modellen mit Differential Privacy die höchste durchschnittliche Genauigkeit von 76,57 Prozent ($\sigma = 5,53\%$) mit einem durchschnittlichen F1-Score von 74,8 Prozent ($\sigma = 5,37\%$). Der Rauschmultiplikator zum Erreichen dieses Epsilons betrug für den angewendeten Datensatz 1,2638. Obwohl das Modell mit einem Epsilon von 0,1 den hier besten Grad der Privatisierung und dem höchsten Rauschmultiplikator von 68.6131 aufwies, übertraf die durchschnittliche Performance die des weniger privaten Modells mit einem Epsilon von 1, mit einem errechneten Rauschmultiplikator von 8.1984. Dieses erreicht lediglich eine durchschnittliche Genauigkeit von 66,22 Prozent ($\sigma = 5,41\%$) und einem durchschnittlichen F1-Score von 49,27 Prozent ($\sigma = 24,3\%$). Das Modell $\varepsilon = 0,1$ hingegen erreichte eine durchschnittliche Genauigkeit von 73,16 Prozent ($\sigma = 7,88\%$) und einen durchschnittlichen F1-Score von 68,77 Prozent ($\sigma = 5,22\%$).

Die zwei besten Ergebnisse aus den Durchläufen mit Differential Privacy haben zum einen ein Modell mit $\varepsilon = 10$ mit einer Genauigkeit von 87,4 Prozent und einem F1-Score von 69,13 Prozent

und zum anderen ein Modell mit $\varepsilon = 0,1$ mit einer Genauigkeit von 86,41 Prozent und einem F1-Score von 66,67 Prozent erreicht.

Verglichen mit der Performance des Modells ohne Differential Privacy ($\varepsilon = \infty$) waren alle Modelle mit Differential Privacy um mindestens 15 Prozentpunkte schlechter. Zusätzlich ist anzumerken, dass die Modelle mit Differential Privacy eine höhere Standardabweichung aufzeigten.

Während der Experimente stellte sich heraus, dass nicht alle Modelle mit Differential Privacy konvergierten. Abhängig vom gewählten Epsilon und dem damit einhergehenden hinzugefügten Rauschen steigt der Anteil der sogenannten Nullmodelle. Nullmodelle sind Modelle, die im Laufe des Trainings nicht in der Lage sind, eine sinnvolle Klassifizierung der Eingabedaten zu erlernen. Infolgedessen geben sie für alle Datenpunkte die gleiche Klassifizierung ab und bieten daher keine brauchbare Vorhersageleistung.

Tabelle 6.1.: Vergleich der Modelle unter Anwendung von Differential Privacy bezüglich des Anteils der Nullmodelle und der besten Performance.

ε	**Anteil der Nullmodelle**	**Beste Modellperformance**	
		acc	**f1**
∞	0 %	92,13 %	92,18 %
10	33 %	87,40 %	81,84 %
1	50 %	79,04 %	71,25 %
0,1	75 %	86,41 %	76,80 %

Die Anteile der Nullmodelle und die beste Modellperformance über alle Durchläufe hinweg in Abhängigkeit von den verschiedenen Epsilons werden in Tabelle 6.1 präsentiert. Hier ist es interessant den F1-Score in Precision und Recall aufzuteilen. In beiden Modellen mit $\varepsilon \leq 1$ wurde ein Recall Wert von 95 bis 100 Prozent erreicht, während sich die Precision bei ungefähr 50 Prozent beläuft. Zusätzlich ist zu beachten, dass die Nullmodelle bei der Berechnung der Performance der Modelle aus Abbiildung 6.5 nicht berücksichtigt wurden, da sie keine sinnvollen Vorhersagen liefern. Die Tabelle gibt dennoch Aufschluss darüber, wie die Wahl des Epsilons die Konvergenz der Modelle beeinflusst.

7. Diskussion

In diesem Kapitel werden die Diskussion der in der Masterarbeit vorgestellten Ergebnisse und Ansätze zur Stresserkennung mittels eines Time-Series-Classification-Transformers und der Anwendung von Differential Privacy diskutiert. Die in den vorangegangenen Kapiteln präsentierten Ergebnisse zeigen vielversprechende Leistungen des TSCT-Modells und seiner Fähigkeit, Stress mit einer hohen Genauigkeit zu erkennen. Die Anwendung von Differential Privacy ermöglicht zudem den Schutz der Privatsphäre der Teilnehmer, ohne die Performance des Modells erheblich zu beeinträchtigen. Im Folgenden werden die Hauptergebnisse der Arbeit kritisch betrachtet und mögliche Verbesserungen und Limitationen diskutiert. Dabei werden sowohl die Ergebnisse im Kontext bestehender Studien betrachtet als auch Kritikpunkte wie die Generalisierbarkeit der Ergebnisse, Interpretierbarkeit des Modells und praktische Anwendbarkeit des Ansatzes erörtert.

Die Ergebnisse aus Kapitel 6.2 zeigen, dass eine Transformer-Architektur in Form eines TSCTs zur Stresserkennung mit Hilfe des WESAD-Datensatzes vielversprechend ist. Durch Anpassung einer Keras-Implementierung konnte ein Modell entwickelt werden, das mit einer durchschnittlichen Genauigkeit von 91,89 Prozent, einem durchschnittlichen F1-Score von 91,61 Prozent und einer kleinen Standardabweichung von unter einem Prozent zuverlässig zu einem guten Ergebnis konvergiert. Dabei ist bemerkenswert, dass das Modell teilweise die Ergebnisse von Arbeiten übertrifft, welche alle Modalitäten, also alle Eingangssignale von allen Sensoren im WESAD-Datensatz, nicht nur die der Smartwatch, zur Klassifizierung berücksichtigen. Der Transformer schlägt alle Ergebnisse von Schmidt et al. [12].

Mit einer maximalen Genauigkeit von 92,13 Prozent und einem F1-Score von 92,18 Prozent erreichte das vorgestellte Modell fast die Ergebnisse von Gil-Martin et al. [11], die eine Genauigkeit von 92,7 Prozent mit einem F1-Score von 92,55 Prozent erzielten. Diese Ergebnisse deuten darauf hin, dass der TSCT-Ansatz effektiv ist.

Allerdings ist zu beachten, dass die Ergebnisse dieser Studie möglicherweise nicht vollständig generalisierbar sind, da die Probanden im WESAD-Datensatz nicht unbedingt repräsentativ sind. Von den 15 Teilnehmern sind 13 Männer. Das durchschnittliche Alter liegt zwischen 25 und 30 Jahren. Für eine umfassendere Evaluierung und Generalisierbarkeit der Ergebnisse wäre es sinnvoll, zukünftige Studien mit größeren und vielfältigeren Datensätzen durchzuführen, die verschiedene Altersgruppen, Geschlechter und ethnische Hintergründe repräsentieren. Darüber hinaus könnten weitere Sensordaten, wie beispielsweise Elektrokardiogramm (EKG) Signale, zur Verbesserung der Klassifikationsgenauigkeit verwendet werden, insbesondere wenn solche Daten in zukünftigen Smartwatch-Generationen verfügbar sind.

Ein weiterer Kritikpunkt ist die Interpretierbarkeit von Transformer-Modellen. Obwohl sie in vielen Bereichen, einschließlich der Zeitreihenklassifikation, beeindruckende Ergebnisse erzielen, sind sie bekanntermaßen schwer zu interpretieren und zu erklären. Diese mangelnde Transparenz kann im medizinischen Kontext problematisch sein, da es für Fachleute, Patienten und Entscheidungsträger wichtig ist, die Gründe für die Modellvorhersagen nachvollziehen zu können. Um dieses Problem zu lösen, könnte zukünftige Forschung darauf abzielen, Methoden zur Verbesserung der Interpretierbarkeit von Transformer-Modellen zu entwickeln, indem beispielsweise Erklärungsansätze wie

LIME (Local Interpretable Model-Agnostic Explanations) [88] oder SHAP (SHapley Additive exPlanations) [89] angewendet werden.

Außerdem konzentriert sich vorliegende Arbeit hauptsächlich auf die Entwicklung und Evaluierung des TSCT-Modells, ohne jedoch ausführlich auf die praktische Umsetzung des Ansatzes in realen Anwendungen einzugehen. Die tatsächliche Anwendung des Modells in der Praxis, beispielsweise durch Integration in Smartwatches oder mobile Apps zur Stressüberwachung und -bewältigung, ist von entscheidender Bedeutung, um den Nutzen des Ansatzes vollständig zu erfassen.

Weiterführend wurde in Kapitel 6.3 untersucht, wie sich Differential Privacy auf die Leistung des TSCT auswirkt. Es hat sich gezeigt, dass Differential Privacy erfolgreich angewendet werden kann, sich allerdings Leistungseinbußen, vor allem im F1-Score, niederschlagen. Bei den am besten trainierten Modellen mit Differential Privacy waren zwar keine großen Einbrüche in der Genauigkeit zu vermerken – lediglich ungefähr vier Prozent – allerdings sackt hierbei der F1-Score, hauptsächlich durch einen niedrigen Precision-Wert (Präzision), um ungefähr 25 Prozentpunkte ein. Durchschnittlich über alle trainierten Modelle mit Differential Privacy hinweg sind Einbußen in beiden Metriken von ungefähr 20 Prozent zu sehen (vgl. Kapitel 6.3.2). Im medizinischen Kontext ist ein hohe Sensitivität, ausgedrückt durch den Recall-Wert, oft wichtiger als ein hohe Präzision, weil die Kosten für falsch-negative Ergebnisse (wenn eine Krankheit nicht erkannt wird, obwohl sie vorhanden ist) in der Regel höher sind als die Kosten für falsch-positive Ergebnisse (wenn eine Krankheit fälschlicherweise diagnostiziert wird, obwohl sie nicht vorhanden ist) [90]. Falsch-negative Ergebnisse können zu einer verspäteten oder fehlenden Behandlung führen, was schwerwiegende Folgen für den Patienten haben kann. Ein hoher Recall-Wert stellt sicher, dass die meisten tatsächlichen Fälle einer Krankheit erkannt werden, auch wenn dabei einige falsch-positive Ergebnisse entstehen.

Es ist anzumerken, dass die Ergebnisse, insbesondere bei einem Epsilon von 0,1, nicht unbedingt repräsentativ sind, da, wie in Kapitel 4.8 beschrieben, zwar 30 Durchläufe für jedes der drei Epsilons durchgeführt wurden, allerdings 75 Prozent der trainierten Modelle Nullmodelle waren (vgl. Tabelle 6.1) und dadurch nicht in die durchschnittliche Gesamtperformance, zu sehen in Abbildung 6.5, eingeflossen sind.

Trotz dieser Einschränkungen ist es interessant, dass das Modell mit einem Epsilon von 0,1 bessere Ergebnisse liefert als das Modell mit einem niedrigeren Rauschmultiplikator und einem Epsilon von 1. Eigentlich kann nämlich erwartet werden, dass ein Modell mit einem Epsilon von 1 besser abschneidet. Dies wirft auch Fragen bezüglich der generellen Vorhersehbarkeit der Modelle auf. Die Tatsache, dass dies nicht der Fall ist, macht es schwieriger, die Performance von privaten Modellen abzuschätzen und den Zusammenhang zur Epsilon-Metrik zu erkennen. Dennoch zeigt dies, dass eine höhere Anforderung an den Datenschutz nicht zwangsläufig zu einer schlechteren Leistung führt, was bedeutet, dass hier ein Modell, welches mehr Differential-Privacy-Kriterien erfüllt, genutzt werden kann, ohne größere Einbrüche in der Performance zu erwarten. Es ist jedoch wichtig, weitere Untersuchungen durchzuführen, um die Abweichungen zwischen den verschiedenen Epsilon-Werten besser zu verstehen und die Vorhersagbarkeit der Ergebnisse zu verbessern.

Zusammenfassend zeigen die Ergebnisse dieser Arbeit, dass die Transformer-Architektur, insbesondere der TSCT, fähig ist, Stress mit hoher Genauigkeit im WESAD-Datensatz zu erkennen. Die Anwendung von Differential Privacy erlaubte zudem den Schutz der Privatsphäre der Teilnehmer,

wobei jedoch die Performance von privaten Modellen und der Zusammenhang zur Epsilon-Metrik schwierig abzuschätzen sind. Insgesamt liefert die Arbeit einen wichtigen Beitrag zur Stresserkennung unter Verwendung von TSCT und Differential Privacy. Trotz der genannten Limitationen und Kritikpunkte demonstriert diese Arbeit das Potenzial dieser Methoden und legt den Grundstein für weitere Untersuchungen in diesem Bereich.

8. Fazit und Ausblick

In diesem letzten Kapitel der Arbeit werden im Fazit zunächst die zentralen Ergebnisse und Erkenntnisse der Arbeit zusammengefasst. Anschließend bietet der Ausblick einen Überblick über mögliche Weiterentwicklungen und zukünftige Forschungsrichtungen in Bezug auf die Stresserkennung mit Hilfe von Smartwatch-Daten. Dabei werden die bisherigen Ergebnisse aufgegriffen und Empfehlungen für Ansätze zur Verbesserung der gewählten Methoden gegeben.

8.1. Fazit

In diesem abschließenden Kapitel werden die zentralen Ergebnisse der Masterarbeit zusammengefasst und die Forschungsfragen aus Kapitel 1.2 beantwortet.

Zunächst hat diese Arbeit gezeigt, dass eine Time-Series-Classification-Transformer-Architektur für die Stresserkennung mit Smartwatch-Daten aus dem WESAD-Datensatz [12] gut geeignet ist. Das entwickelte Modell erreichte eine durchschnittliche Genauigkeit von 91,89 Prozent – mit einem Maximum von 92,13 Prozent – und wies dabei eine kleine Standardabweichung von unter einem Prozent auf, was auf eine zuverlässige und konsistente Leistung hindeutet. Schon der erste Prototyp basierend auf der ursprünglichen Konfiguration der Architektur und Hyperparameter aus dem Implementierungsbeispiel von Keras [75] wies für vorliegendes Problem eine Genauigkeit von über 87 Prozent auf. Mit Hilfe weiterer Anpassungen der Architektur und Hyperparameter-Optimierung konnte das Modell um weitere fünf Prozentpunkte verbessert werden.

Weiterhin wurde das TSCT-Modell mit den Ergebnissen von Gil-Martin et al. [11] verglichen, deren Signalvorverarbeitung als Grundlage für vorliegende Arbeit diente. Die erzielten Ergebnisse des TSCT-Modells lagen nahezu auf dem Niveau der CNN-Architektur von Gil-Martin et al. Dies deutet darauf hin, dass der TSCT-Ansatz effektiv ist und mit weiteren Optimierungen möglicherweise noch bessere Ergebnisse erzielt werden können.

Schlussendlich wurde in dieser Arbeit untersucht, welche Auswirkungen Differential Privacy auf die Leistung des TSCT-Modells hat. Die Ergebnisse zeigen, dass Differential Privacy erfolgreich angewendet werden kann, ohne die Modellleistung drastisch zu beeinträchtigen. Bei den besten trainierten Modellen mit Differential Privacy und einem Epsilon von 0,1, was für einen ausreichenden Schutz gegen datenschutzgefährdende Angriffe auf Datensätze spricht [30], war lediglich eine geringe Abnahme der Genauigkeit um etwa vier Prozent, allerdings große Einbußen im F1-Score von ungefähr 25 Prozent, festzustellen.

Zusammenfassend hat diese Masterarbeit wichtige Erkenntnisse zur Anwendung von Transformer-Architekturen, insbesondere der Time-Series-Classification-Transformer, in der Stresserkennung mittels Smartwatch-Daten gewonnen. Darüber hinaus wurden die Auswirkungen von Differential Privacy auf die Modellleistung untersucht, wobei gezeigt wurde, dass Transformer auch bei datenschutzkritischen Anwendungen eingesetzt werden können. Insgesamt eröffnen die Ergebnisse dieser Arbeit neue Perspektiven für zukünftige Forschung und Anwendungen im Bereich der Stresserkennung und des Datenschutzes.

8.2. Ausblick

Im finalen Unterkapitel dieser Arbeit wird ein Ausblick auf mögliche Weiterentwicklungen und zukünftige Forschungsrichtungen im Bereich der Stresserkennung mit Smartwatch-Daten gegeben. Dabei wird auf die Ergebnisse der vorliegenden Arbeit Bezug genommen und Empfehlungen für Ansätze zur Verbesserung der gewählten Methoden gegeben.

Ein wesentlicher Aspekt für zukünftige Forschungen ist die Erweiterung der verwendeten Sensordaten. In dieser Arbeit wurden ausschließlich Daten von der medizinischen Smartwatch E4 verwendet. Obwohl dieses Gerät für den Zeitpunkt der Studie von Schmidt et al. [12] bereits viele Sensoren enthielt, verfügen neuere Modelle, wie beispielsweise die Apple Smartwatch, über zusätzliche Sensoren wie einen EKG-Sensor. Die Integration von EKG-Daten in das Training von Maschinenlernmodellen kann, wie die Ergebnisse von Gil-Martin et al. [11] zeigen, zu einer deutlichen Verbesserung der Performance führen.

Ein weiterer wichtiger Faktor für die Verbesserung der Modellperformance ist die Größe des Datensatzes. Die Vorteile von Transformer-Architekturen gegenüber herkömmlichen RNN-Netzen, wie LSTM-Netzen, werden mit zunehmender Größe des Datensatzes immer deutlicher. Da der WESAD-Datensatz lediglich 15 Teilnehmer umfasst, ist es wahrscheinlich, dass eine Erweiterung des Datensatzes zu einer besseren Performance führt. Eine solche Erweiterung kann durch zusätzliche reale Experimente oder die Synthese von Daten mit Hilfe von Generative Adversarial Networks (GANs) erreicht werden.

In Bezug auf mögliche Verbesserungen des Modells könnten weitere Experimente mit der Hyperparameter-Optimierung durchgeführt oder zusätzliche Schichten wie CNN- oder LSTM-Schichten hinzugefügt werden. Es ist auch interessant zu untersuchen, ob die Signalverarbeitung von Gil-Martin et al., die für die Nutzung einer CNN-Architektur vorteilhaft war, für einen Transformer tatsächlich notwendig ist oder ob eine Rohform des Datensatzes sogar zu besseren Ergebnissen führen kann.

Bei der Anwendung von Differential Privacy auf den TSCT könnten zusätzliche Experimente durchgeführt werden, um die Hyperparameter für das spezifische Epsilon noch einmal anzupassen und die Modellperformance zu steigern. Möglicherweise können auf diese Weise noch bessere Ergebnisse erzielt werden.

Insgesamt zeigt die vorliegende Masterarbeit, dass es noch viel Raum für Verbesserungen und Weiterentwicklungen im Bereich der Stresserkennung mittels Smartwatch-Daten gibt. Durch die Integration von neuen Sensordaten, die Erweiterung der Datenbasis und die kontinuierliche Optimierung von Modellen und Techniken kann zukünftige Forschung dazu beitragen, die Klassifikationsleistung und Generalisierbarkeit der Ergebnisse kontinuierlich zu steigern.

Literatur

[1] Yaribeygi H et al. „The impact of stress on body function: A review.“ In: *EXCLI Journal* 16 (2017), S. 1057–1072.

[2] Miller GE Segerstrom SC. „Psychological stress and the human immune system: a meta-analytic study of 30 years of inquiry.“ In: *Psychol Bull.* 130 (2004), S. 601–30.

[3] Miller GE Cohen S Janicki-Deverts D. „Psychological stress and disease.“ In: *JAMA.* 298 (2007), S. 1685–7.

[4] Kivimäki M. Steptoe A. „Stress and cardiovascular disease: an update on current knowledge.“ In: *Annu Rev Public Health.* 34 (2013), S. 337–54.

[5] H.M. van Praag. „Can stress cause depression?“ In: *Progress in Neuro-Psychopharmacology and Biological Psychiatry* 28.5 (2004), S. 891–907. ISSN: 0278-5846. DOI: https://doi.org/10.1016/j.pnpbp.2004.05.031. URL: https://www.sciencedirect.com/science/article/pii/S0278584604000892.

[6] Alexandros M. Heraclides u. a. „Work Stress, Obesity and the Risk of Type 2 Diabetes: Gender-Specific Bidirectional Effect in the Whitehall II Study“. In: *Obesity* 20.2 (2012), S. 428–433.

[7] Analava Mitra. „Diabetes and Stress: A Review“. In: *Ethno-Med* 2282657 (Juli 2008), S. 131–135. DOI: 10.1080/09735070.2008.11886324.

[8] S Michie. „Causes and Management of stress at work“. In: *Occupational and Environmental Medicine* 59.1 (2002), S. 67–72. ISSN: 1351-0711. DOI: 10.1136/oem.59.1.67. eprint: https://oem.bmj.com/content/59/1/67.full.pdf. URL: https://oem.bmj.com/content/59/1/67.

[9] Giorgos Giannakakis u. a. „Review on Psychological Stress Detection Using Biosignals“. In: *IEEE Transactions on Affective Computing* 13.1 (2022), S. 440–460. DOI: 10.1109/TAFFC.2019.2927337.

[10] Rubén San-Segundo u. a. „Robust Human Activity Recognition Using Smartwatches and Smartphones“. In: *Eng. Appl. Artif. Intell.* 72 (2018), S. 190–202. DOI: 10.1016/j.engappai.2018.04.002. URL: https://doi.org/10.1016/j.engappai.2018.04.002.

[11] Manuel Gil-Martin u. a. „Human Stress Detection With Wearable Sensors Using Convolutional Neural Networks“. In: *IEEE Aerospace and Electronic Systems Magazine* 37.1 (2022), S. 60–70. DOI: 10.1109/MAES.2021.3115198.

[12] Philip Schmidt u. a. „Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection“. In: ICMI '18. Boulder, CO, USA: Association for Computing Machinery, 2018, S. 400–408. ISBN: 9781450356923. DOI: 10.1145/3242969.3242985. URL: https://doi.org/10.1145/3242969.3242985.

[13] Ashish Vaswani u. a. „Attention is All you Need“. In: *Advances in Neural Information Processing Systems*. Hrsg. von I. Guyon u. a. Bd. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[14] Hans Selye. „The general adaptation syndrome and the diseases of adaptation“. In: *The journal of clinical endocrinology* 6.2 (1946), S. 117–230.

[15] Cornelia Setz u. a. „Discriminating stress from cognitive load using a wearable EDA device“. In: *IEEE Transactions on information technology in biomedicine* 14.2 (2009), S. 410–417.

[16] Eugenijus Kaniusas und Eugenijus Kaniusas. „Fundamentals of biosignals“. In: *Biomedical Signals and Sensors I: Linking Physiological Phenomena and Biosignals* (2012), S. 1–26.

[17] A Moriguchi u. a. „Spectral change in heart rate variability in response to mental arithmetic before and after the beta-adrenoceptor blocker, carteolol“. In: *Clinical Autonomic Research* 2 (1992), S. 267–270.

[18] Vesna Vuksanović und Vera Gal. „Heart rate variability in mental stress aloud“. In: *Medical engineering & physics* 29.3 (2007), S. 344–349.

[19] Els Clays u. a. „The perception of work stressors is related to reduced parasympathetic activity“. In: *International archives of occupational and environmental health* 84 (2011), S. 185–191.

[20] ME Dawson, AM Schell und DL Filion. *The electrodermal system. Handbook of psychophysiology 2, 159–181*. 2007.

[21] Giorgos Giannakakis u. a. „Review on Psychological Stress Detection Using Biosignals“. In: *IEEE Transactions on Affective Computing* 13.1 (2022), S. 440–460. DOI: 10.1109/TAFFC.2019.2927337.

[22] Dzmitry Bahdanau, Kyunghyun Cho und Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. DOI: 10.48550/ARXIV.1409.0473. URL: https://arxiv.org/abs/1409.0473.

[23] U. Kamath, K.L. Graham und W. Emara. *Transformers for Machine Learning: A Deep Dive*. CRC Press, 2022, S. 11–12. ISBN: 9781000587098. URL: https://books.google.de/books?id=zENpEAAAQBAJ.

[24] Hao Jiang, Lianguang Liu und Cheng Lian. „Multi-Modal Fusion Transformer for Multivariate Time Series Classification“. In: *2022 14th International Conference on Advanced Computational Intelligence (ICACI)*. 2022, S. 284–288. DOI: 10.1109/ICACI55529.2022.9837525.

[25] Huiling Chen u. a. „Early Time Series Classification Using TCN-Transformer“. In: *2022 IEEE 4th International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*. 2022, S. 1079–1082. DOI: 10.1109/ICCASIT55263.2022.9986835.

[26] Chao Zhang u. a. „Soft Sensing Transformer: Hundreds of Sensors are Worth a Single Word“. In: *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, Dez. 2021. DOI: 10.1109/bigdata52589.2021.9671925. URL: https://doi.org/10.1109%2Fbigdata52589.2021.9671925.

[27] Qingsong Wen u. a. *Transformers in Time Series: A Survey*. 2022. DOI: 10.48550/ARXIV.2202.07125. URL: https://arxiv.org/abs/2202.07125.

[28] Cynthia Dwork. „Differential Privacy: A Survey of Results“. In: *Theory and Applications of Models of Computation*. Hrsg. von Manindra Agrawal u. a. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 1–19. ISBN: 978-3-540-79228-4.

[29] Lucas Lange u. a. *Privacy in Practice: Private COVID-19 Detection in X-Ray Images*. 2022. DOI: 10.48550/ARXIV.2211.11434. URL: https://arxiv.org/abs/2211.11434.

[30] Milad Nasr u. a. *Adversary Instantiation: Lower Bounds for Differentially Private Machine Learning*. 2021. DOI: 10.48550/ARXIV.2101.04535. URL: https://arxiv.org/abs/2101.04535.

[31] Duy Vu und Aleksandra Slavkovic. „Differential privacy for clinical trial data: Preliminary evaluations“. In: *2009 IEEE International Conference on Data Mining Workshops*. IEEE. 2009, S. 138–143.

[32] Fida Kamal Dankar und Khaled El Emam. „Practicing differential privacy in health care: A review.“ In: *Trans. Data Priv.* 6.1 (2013), S. 35–67.

[33] Florian Tramèr u. a. „Differential privacy with bounded priors: reconciling utility and privacy in genome-wide association studies“. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, S. 1286–1297.

[34] Alexander Ziller u. a. „Medical imaging deep learning with differential privacy“. In: *Scientific Reports* 11.1 (2021), S. 1–8.

[35] Zhihan Lv und Francesco Piccialli. „The security of medical data on internet based on differential privacy technology“. In: *ACM Transactions on Internet Technology* 21.3 (2021), S. 1–18.

[36] Xinyi Li u. a. *DP-LSTM: Differential Privacy-inspired LSTM for Stock Prediction Using Financial News*. 2019. DOI: 10.48550/ARXIV.1912.10806. URL: https://arxiv.org/abs/1912.10806.

[37] Muneeb Ul Hassan, Mubashir Husain Rehmani und Jinjun Chen. „Differential privacy in blockchain technology: A futuristic approach“. In: *Journal of Parallel and Distributed Computing* 145 (2020), S. 50–74.

[38] Steven Ruggles u. a. „Differential privacy and census data: Implications for social and economic research“. In: *AEA papers and proceedings*. Bd. 109. American Economic Association 2014 Broadway, Suite 305, Nashville, TN 37203. 2019, S. 403–408.

[39] Qian Wang u. a. „Real-Time and Spatio-Temporal Crowd-Sourced Social Network Data Publishing with Differential Privacy“. In: *IEEE Transactions on Dependable and Secure Computing* 15.4 (2018), S. 591–606. DOI: 10.1109/TDSC.2016.2599873.

[40] Liyue Fan. „Image pixelization with differential privacy“. In: *Data and Applications Security and Privacy XXXII: 32nd Annual IFIP WG 11.3 Conference, DBSec 2018, Bergamo, Italy, July 16–18, 2018, Proceedings 32*. Springer. 2018, S. 148–162.

[41] Daniel L Oberski und Frauke Kreuter. „Differential privacy and social science: An urgent puzzle“. In: *Harvard Data Science Review* 2.1 (2020), S. 1.

[42] Christine Task und Chris Clifton. „A guide to differential privacy theory in social network analysis“. In: *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE. 2012, S. 411–417.

[43] Zhan Qin u. a. „Generating synthetic decentralized social graphs with local differential privacy“. In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2017, S. 425–438.

[44] Haiping Huang u. a. „Privacy-preserving approach PBCN in social network with differential privacy“. In: *IEEE Transactions on Network and Service Management* 17.2 (2020), S. 931–945.

[45] Sepp Hochreiter und Jürgen Schmidhuber. „Long Short-Term Memory“. In: *Neural Comput.* 9.8 (Nov. 1997), S. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[46] Junyoung Chung u. a. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. DOI: 10.48550/ARXIV.1412.3555. URL: https://arxiv.org/abs/1412.3555.

[47] Pranav Rajpurkar u. a. *Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks*. 2017. DOI: 10.48550/ARXIV.1707.01836. URL: https://arxiv.org/abs/1707.01836.

[48] Chao Che u. a. „Constrained transformer network for ECG signal processing and arrhythmia classification“. In: *BMC Medical Informatics and Decision Making* 21 (Juni 2021). DOI: 10.1186/s12911-021-01546-2.

[49] Saeed Saadatnejad, Mohammadhosein Oveisi und Matin Hashemi. „LSTM-Based ECG Classification for Continuous Monitoring on Personal Wearable Devices“. In: *IEEE Journal of Biomedical and Health Informatics* 24.2 (2020), S. 515–523. DOI: 10.1109/JBHI.2019.2911367.

[50] Albert Zeyer u. a. „A Comparison of Transformer and LSTM Encoder Decoder Models for ASR“. In: *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2019, S. 8–15. DOI: 10.1109/ASRU46091.2019.9004025.

[51] Behnam Behinaein u. a. „A Transformer Architecture for Stress Detection from ECG“. In: *2021 International Symposium on Wearable Computers*. ISWC '21. Virtual, USA: Association for Computing Machinery, 2021, S. 132–134. ISBN: 9781450384629. DOI: 10.1145/3460421.3480427. URL: https://doi.org/10.1145/3460421.3480427.

[52] Ruiqi Wang u. a. *Husformer: A Multi-Modal Transformer for Multi-Modal Human State Recognition*. 2022. DOI: 10.48550/ARXIV.2209.15182. URL: https://arxiv.org/abs/2209.15182.

[53] Prerna Garg u. a. „Stress detection by machine learning and wearable sensors“. In: *26th International Conference on Intelligent User Interfaces-Companion*. 2021, S. 43–45.

[54] Pramod Bobade und M. Vani. „Stress Detection with Machine Learning and Deep Learning using Multimodal Physiological Data“. In: *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*. 2020, S. 51–57. DOI: 10.1109/ICIRCA48905.2020.9183244.

[55] Dhananjai Bajpai und Lili He. „Evaluating KNN performance on WESAD dataset“. In: *2020 12th International Conference on Computational Intelligence and Communication Networks (CICN)*. IEEE. 2020, S. 60–62.

[56] Salar Jafarlou u. a. „ECG Biosignal Deidentification Using Conditional Generative Adversarial Networks“. In: *2022 44th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2022, S. 1366–1370. DOI: 10.1109/EMBC48229.2022.9872015.

[57] Alfredo J. Perez und Sherali Zeadally. „Privacy Issues and Solutions for Consumer Wearables“. In: *IT Professional* 20.4 (2018), S. 46–56. DOI: 10.1109/MITP.2017.265105905.

[58] Ashkan Yousefpour u. a. „Opacus: User-Friendly Differential Privacy Library in PyTorch“. In: *CoRR* abs/2109.12298 (2021). arXiv: 2109.12298. URL: https://arxiv.org/abs/2109.12298.

[59] Muneeb Ul Hassan, Mubashir Husain Rehmani und Jinjun Chen. „Differential privacy techniques for cyber physical systems: a survey“. In: *IEEE Communications Surveys & Tutorials* 22.1 (2019), S. 746–789.

[60] Jiajun Zhang u. a. *Re-DPoctor: Real-time health data releasing with w-day differential privacy*. 2017. DOI: 10.48550/ARXIV.1711.00232. URL: https://arxiv.org/abs/1711.00232.

[61] Thông T. Nguyên u. a. *Collecting and Analyzing Data from Smart Device Users with Local Differential Privacy*. 2016. DOI: 10.48550/ARXIV.1606.05053. URL: https://arxiv.org/abs/1606.05053.

[62] Nicolas Papernot u. a. „Tempered Sigmoid Activations for Deep Learning with Differential Privacy“. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.10 (Mai 2021), S. 9312–9321. DOI: 10.1609/aaai.v35i10.17123. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17123.

[63] Martin Abadi u. a. „Deep Learning with Differential Privacy“. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Okt. 2016. DOI: 10.1145/2976749.2978318. URL: https://doi.org/10.1145%2F2976749.2978318.

[64] Eugene Bagdasaryan, Omid Poursaeed und Vitaly Shmatikov. „Differential Privacy Has Disparate Impact on Model Accuracy“. In: *Advances in Neural Information Processing Systems*. Hrsg. von H. Wallach u. a. Bd. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/fc0de4e0396fff257ea362983c2dda5a-Paper.pdf.

[65] *Keras: Deep Learning for humans*. https://github.com/keras-team/keras. Aufgerufen: 27.03.2023.

[66] Thomas Kluyver u. a. *Jupyter Notebooks-a publishing format for reproducible computational workflows*. Bd. 2016. 2016.

[67] Yann LeCun. „1.1 Deep Learning Hardware: Past, Present, and Future“. In: *2019 IEEE International Solid- State Circuits Conference - (ISSCC)*. 2019, S. 12–19. DOI: 10.1109/ISSCC.2019.8662396.

[68] Ekaba Bisong und Ekaba Bisong. „Google colaboratory“. In: *Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners* (2019), S. 59–64.

[69] Sabyasachi Chakraborty u. a. „A Multichannel Convolutional Neural Network Architecture for the Detection of the State of Mind Using Physiological Signals from Wearable Devices“. In: *Journal of Healthcare Engineering* 2019 (Okt. 2019). DOI: 10.1155/2019/5397814.

[70] Rich Caruana und Alexandru Niculescu-Mizil. „An empirical comparison of supervised learning algorithms“. In: *Proceedings of the 23rd international conference on Machine learning*. 2006, S. 161–168.

[71] David H Wolpert. „The lack of a priori distinctions between learning algorithms“. In: *Neural computation* 8.7 (1996), S. 1341–1390.

[72] Pavel Brazdil u. a. *Metalearning: Applications to data mining*. Springer Science & Business Media, 2008.

[73] Matthias Feurer u. a. „Efficient and robust automated machine learning“. In: *Advances in neural information processing systems* 28 (2015).

[74] Randal S Olson und Jason H Moore. „TPOT: A tree-based pipeline optimization tool for automating machine learning“. In: *Workshop on automatic machine learning*. PMLR. 2016, S. 66–74.

[75] Theodoros Ntakouris. *Timeseries classification with a Transformer model*. `https://keras.io/examples/timeseries/timeseries_transformer_classification/`. Aufgerufen: 05.03.2023.

[76] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[77] Christopher M Bishop und Nasser M Nasrabadi. *Pattern recognition and machine learning*. Bd. 4. 4. Springer, 2006.

[78] Tom Fawcett. „An introduction to ROC analysis“. In: *Pattern recognition letters* 27.8 (2006), S. 861–874.

[79] John Paul Varkey, Dario Pompili und Theodore A Walls. „Human motion recognition using a wireless sensor-based wearable system“. In: *Personal and Ubiquitous Computing* 16 (2012), S. 897–910.

[80] Wolfram Boucsein. *Electrodermal activity*. Springer Science & Business Media, 2012.

[81] Lutz Prechelt. „Early stopping-but when?“ In: *Neural Networks: Tricks of the trade*. Springer, 2002, S. 55–69.

[82] Tong Zhang und Bin Yu. „Boosting with early stopping: Convergence and consistency“. In: (2005).

[83] Milad Nasr u. a. *Adversary Instantiation: Lower Bounds for Differentially Private Machine Learning*. 2021. arXiv: `2101.04535 [cs.LG]`.

[84] Nicholas Carlini u. a. „The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks.“ In: *USENIX Security Symposium*. Bd. 267. 2019.

[85] Kaiming He u. a. *Deep Residual Learning for Image Recognition*. 2015. arXiv: `1512.03385 [cs.CV]`.

[86] David MW Powers. „Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation“. In: *arXiv preprint arXiv:2010.16061* (2020).

[87] Cynthia Dwork. „Differential privacy“. In: *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II 33*. Springer. 2006, S. 1–12.

[88] Saumitra Mishra, Bob L Sturm und Simon Dixon. „Local interpretable model-agnostic explanations for music content analysis.“ In: *ISMIR*. Bd. 53. 2017, S. 537–543.

[89] Sujith Mangalathu, Seong-Hoon Hwang und Jong-Su Jeon. „Failure mode and effects analysis of RC members based on machine-learning-based SHapley Additive exPlanations (SHAP) approach“. In: *Engineering Structures* 219 (2020), S. 110927.

[90] Rajul Parikh u. a. „Understanding and using sensitivity, specificity and predictive values“. In: *Indian journal of ophthalmology* 56.1 (2008), S. 45.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit mit dem Thema:

„Entwicklung eines Privatsphäre-erhaltenden Transformers zur Stresserkennung anhand von Smartwatch-Health-Daten“

selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.

Leipzig, den 17.04.2023

_____

Borislav Degenkolb