Advances in Intelligent Systems

Dozierender: Professor Dr. Dennis Müller

Projekt: AI-based ECG-Analysis

Von: Miriam Agrawala

Einleitung

Herz-Kreislauf-Erkrankungen zählen weltweit zu den häufigsten Krankheiten. In den westlichen Ländern sind sie mit rund 45% und in Entwicklungsländern 25% aller Todesfälle die häufigste Todesursache. Weltweit sterben jährlich rund 17,3 Millionen Menschen an den Folgen einer Herz-Kreislauf-Erkrankung, wobei Schätzungen einen Anstieg auf bis zu 23,6 Millionen im Jahr 2030 prognostizieren. (1) Eine der größten Herausforderungen heutzutage sind deshalb die Prävention, frühe Diagnose und das Management dieser Erkrankungen.
Das Elektrokardiogramm ist für die Diagnose von Herzerkrankungen weit verbreitet. Seine Auswertung ist jedoch zeitaufwendig und fehleranfällig.
In der folgenden Studie soll sich deshalb mit der KI-gestützten Verarbeitung von EKGs beschäftigt werden. Es wird eine Übersicht über den aktuellen Stand der Forschung gegeben, daraufhin werden die in dieser Studie verwendeten Methoden und Ergebnisse vorgestellt. Es folgt eine Anleitung zur Verwendung des zur Verfügung gestellten Codes. Am Ende steht der Ausblick über die Fortführung des Projektes.

Hintergrundwissen: Elektrokardiogramm

Ein wichtiges diagnostisches Mittel zur Diagnose von Herzerkrankungen ist das Elektrokardiogramm (EKG). Hierbei handelt es sich um die Aufzeichnung der elektrischen Aktivität von Herzmuskelfasern. Es ist eine ausgereifte, im medizinischen Alltag etablierte Methode, die vor allem in der Allgemein- und Notfallmedizin einen hohen Stellenwert hat. Vor allem bei akuten Herzinfarkten und Herzrhythmusstörungen wird es angewendet.
Es gibt verschiedenste Apparate zur Aufzeichnung von EKGs, abhängig von den jeweiligen Anwendungsszenarien. So nimmt auch die Integration von EKG-Funktionen in Wearables wie Smartwatches zu.

Die Pumpfunktion des Herzens ist abhängig von der elektrischen Erregung des Sinusknoten. Dabei handelt es sich um eine Ansammlung von Nervengewebe in der Herzmuskulatur, die dem Herz den elektrischen Impuls gibt, der für jeden einzelnen Schlag notwendig ist. Vom Sinusknoten ausgehend breitet sich der elektrische Impuls über das Erregungsleitungssystem des Herzens aus. Die dabei entstehenden Potentialänderung kann man mit EKG-Elektroden an der Körperoberfläche abgreifen und in Bezug zur Zeit aufzeichnen. Aus diesen Messungen können Rückschlüsse auf die Gesundheit des Herzens gezogen werden. Am typischsten im klinischen Alltag ist das 12-Kanal EKG, bei dem die Ströme über 12 Elektroden gemessen werden. Es resultieren 12 verschiedene Kurven (s. Abb. 2).

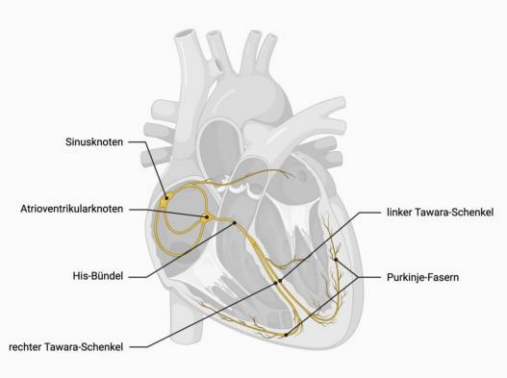

Abb. 1: Nervales Erregungsleitungssystem des Herzenz. Die elektrische Erregung breitet sich ausgehend vom Sinusknoten über den Rest des Herzens aus.

Abb. 2: 12-Kanal EKG. Die Messung jedes Kanals wird durch eine andere Elektrode an der Körperoberfläche aufgezeichnet.

Es gibt zahlreiche Indikationen, wegen denen ein EKG durchgeführt werden kann. Hierzu gehören unter anderem die Herzinsuffizienz, Herzrythmusstörungen („Herzrasen“, „Herzstolpern“), Herzinfarkte, aber auch der Verdacht auf Vergiftungen oder Lungenerkrankungen. (2)

Einsatz von KI in der EKG-Analyse: State oft the Art

Die Analyse von EKGs ist für ÄrztInnen sehr zeitaufwendig. Zunehmend wird deshalb versucht, die Auswertung durch Algorithmen der Künstlichen Intelligenz vorzunehmen. Dabei können diese Algorithmen unterstützen bei der Interpretation und Erkennung von EKG-Befunden, aber auch Messungen in Herzschrittmachern auswerten. Weiterhin können Sie verwendet werden, um die Qualität von EKG-Messungen durch Entfernung von Hintergrundrauschen zu verbessern. (3)

Die KI-unterstützte EKG-Interpretation hat bereits vielversprechende Ergebnisse hervorgebracht. Der State oft the Art soll im Folgenden vorgestellt werden.

Zhang et al. schlagen ein „Attention-based“ Convolutional Recurrent Neural Network vor („STA-CRNN“) um EKG-Signale aus 12-Kanal EKGs mit einer Länge zwischen 6 und 60 Sekunden zu klassifizieren. In diesem werden Convolutional Neural Networks verwendet, um Features aus den EKG-Messungen zu extrahieren. Zusätzlich werden Recurrent Neural Networks eingesetzt, um die zeitliche Komponente des EKGs zu verarbeiten. Da die verschiedenen Kanäle und zeitlichen Segmente der Feature Map, die aus dem CNN resultiert, unterschiedlich stark zu der Vorhersage von EKG-Signalen beitragen, wurde ein „Attention Mechanism“ zu der Architektur hinzugefügt. Durch diesen soll sich das Netz besser auf ausschlaggebende Features konzentrieren können. Zhang et al. erreichen mit dieser Architektur einen F1-Score von 0,835 bei der Klassifizierung von neun verschiedenen Herzrhythmen. Durch eine Visualisierung können Zhang et al. außerdem zeigen, dass die von STA-CRNN gelernten Features denen entsprechen, die auch ÄrztInnen für die Einschätzung von EKGs verwenden würden. (4)

Chang et al. verwendeten eine LSTM-Architektur um 12 unterschiedliche Herzrhythmen aus 12-Kanal EKGs vorherzusagen. Die erreichte Accuracy war bei 0,982, mit einer ROC von 0,987, der F1 Score über 0,777. Es wurde gezeigt, dass die Accuracy des Modells der von Ärztinnen und Ärzten überlegen war: Internisten erreichten 0,55, Notärzte 0,73 und Kardiologen 0,83. (5)

Weiterhin wurde eine KI-Architektur vorgeschlagen, um mit einer Kombination aus CNN und LSTMs fünf verschiedene Klassen von Herzrhythmen aus Aufnahmen verschiedener Länge vorherzusagen. Mit verschiedenen Architekturen wurden Accuracies von 98,10% (6), bzw. 99,85% (7) erreicht.

Ein weiterer Ansatz ist die Verwendung einer Kombination aus Wavelet-Paket-Transformation (eine Methode aus der digitalen Signalverarbeitung zur Analyse und Kompression digitaler Signale) mit Rekonstruktion der EKG-Signale. Aus den rekonstruierten EKG-Signalen werden Feature Vektoren gebildet. Diese werden mit einem k-nearest-neighbour Classifier automatisch klassifiziert. Dabei wurde eine hohe diagnostische Sensitivität und Spezifität von jeweils 99,64% bzw. 99,71% erreicht, außerdem eine Accuracy von 99,65%. (8)

Da Support Vector Machines sehr effektiv in hoch dimensionalen Datensätzen funktionieren, gab es auch Ansätze, um mit dieser Technologie EKG-Daten zu analysieren. Bei der Analyse von 2-minütigen EKGs, um die Aktivität vorherzusagen, die die Testperson gerade ausführt, wurde eine Accuracy von 88,49% erreicht. (9) Bei der Verwendung von SVM, um aus EKGs verschiedene Krankheitsbilder vorherzusagen, wurde eine Accuracy von 99,2% erreicht. (10)

Hintergrundwissen: LSTM und CNN

Long Short-Term Memory (LSTM) Modelle sind eine Form der Recurrent Neural Networks (RNN). Es wird vor allem für die Mustererkennung in Sequenzen von Daten verwendet, beispielsweise in Aktienkursen oder natürlicher Sprache. Es ist besonders nützlich, wenn es darum geht, lange Abhängigkeiten innerhalb von Sequenzen zu erfassen, da es Mechanismen zur Speicherung von Informationen über lange Zeitintervalle hinweg hat. (11)
LSTM-Einheiten besitzen die Fähigkeit, Informationen selektiv zu behalten oder zu vergessen, indem sie ein Langzeitgedächtnis aufrechterhalten. Informationen von hoher Bedeutung werden behalten und rückwärts propagiert, während unwichtige Informationen vergessen und verworfen werden. (6)

Convolutional Neural Networks (CNNs) verwenden Faltungs- und Poolingoperationen, um die Größe der Daten zu reduzieren und wichtige Merkmale zu erfassen. Dies macht sie besonders effektiv für Aufgaben wie Bilderkennung oder -segmentierung. (12) Durch ihre Fähigkeit, Positionen von Details auf Bildern zu bestimmen und translationsinvariante Muster zu erkennen, werden CNNs auch vermehrt auf physiologische Signale angewendet. (13)

Methoden: Datensatz

Für die diese Studie wurde der Datensatz „PTB-XL“ verwendet, der von Wagner et al. im Jahr 2022 veröffentlicht wurde. (14) (15)
Es handelt sich um einen Datensatz aus 21799 12-Kanal EKGs, die an 18869 PatientInnen (52% männlich, 48% weiblich, Alter 0-95 Jahre) zwischen 1989 und 1996 gemessen wurden. Jede Messung ist 10 Sekunden lang.
Die Rohdaten wurden entweder von zwei KardiologInnen oder automatisch durch das EKG-Gerät annotiert, wobei die automatisch annotierten Daten von einem Kardiologen/ einer Kardiologin validiert wurden. Die Benennung erfolgte nach dem sogenannten SCP-ECG Standard, welcher die Computergestützte Verarbeitung von EKGs vereinheitlichen soll. Jeder Messung wurden verschiedene Diagnosen zugeordnet, zusammen mit einer Prozentzahl, die ausdrückt, wie wahrscheinlich die jeweilige Diagnose ist. In dieser Studie wurde jeweils die als wahrscheinlichste angegebene Diagnose als Label verwendet.

Die Diagnosen sind wie folgt im Datensatz verteilt:

#Records	Superclass	Description
9514	NORM	Normal ECG
5469	MI	Myocardial Infarction
5235	STTC	ST/T Change
4898	CD	Conduction Disturbance
2649	HYP	Hypertrophy

Abb. 3: Anzahl der Samples in den verschiedenen diagnostischen Klassen des Datensatzes. Diese dienen dem späteren Modelltraining als Label.

Die Diagnosen wurden in fünf verschieden Klassen eingeordnet:

- Normale EKGs
- „Myocardial infarction“: Herzinfarkt
- „ST/T Change“: Veränderung eines bestimmten EKG-Abschnitts, der z.B. mit einer speziellen Art des Herzinfarkts zusammenhängen kann
- „Conduction Disturbance“: Leitungsstörungen des Herzens, wie Arrhythmien des Herzschlags
- „Hypertrophy“: Vergrößerung des Herzmuskels

Diesen fünf Superdiagnosen wurden 71 verschiedene Subdiagnosen zugeordnet, wobei in dieser Studie nur die Superdiagnosen als Labels verwendet wurden.

Der Datensatz enthält weitere Informationen, zum Beispiel das Alter der PatientInnen, den Namen des EKG-Gerät, das zur Aufnahme verwendet wurde oder auch detaillierte kardiologische Informationen.

In dieser Studie wurden die EKG-Aufnahmen als Features verwendet. Die dazugehörigen Diagnosen waren die Label.

Jede Messung steht mit einer Abtastrate von 100Hz und einer Abtastrate von 500Hz zur Verfügung. Für diese Studie wurden die 100Hz-Messungen verwendet, wodurch jedes Sample ein Format von 1000 Messpunkten auf 12 Kanälen hat.

Aus dem PTB-XL Datensatz wurden für die vorliegende Studie drei verschiedene Datensätze gebildet.

Ein „Standarddatensatz“, in dem jedes Sample die volle Länge von 1000 Messpunkten auf 12 Kanälen hat.

Weiterhin wurde, um ein robusteres Training zu erreichen, ein Datensatz erstellt, bei dem unter den ersten 200 Messpunkten zufällig einer als Startpunkt gewählt wurde. Ab diesem zufälligen Punkt wurde für ein Sample eine Länge von 800 Messpunkten verwendet, wieder auf 12 Kanälen. Durch diese Randomisierung wurde versucht, eine Datenaugmentation zu erreichen, um das Training robuster zu gestalten.

Schließlich wurde ein Datensatz erstellt, der nur aus den letzten 200 Messpunkten des Originaldatensatzes bestand. Auf diese Weise sollte getestet werden, ob eine Sequenzlänge von 1000 Messpunkten die Kapazität des Long Term Memory eines LSTMs überschreitet und es deshalb für die Klassifizierung möglicherweise keinen Unterschied machen würde, dass man nur die letzten 200 Messpunkte verwendet.

Methoden: Modell-Architekturen und Modifikationen

Es gab fünf verschiedene Architekturen von neuronalen Netzen, bei denen es sich um LSTMs oder Verbindungen von LSTMs mit CNNs handelte.

Die erste Architektur war ein Long Short Term Memory Netzwerk mit einer einzelnen LSTM-Schicht. Weiterhin wurden „stacked“ LSTMs getestet, bei denen mehrere LSTM-Schichten aneinandergereiht

werden.
Zunehmende Tiefe von neuronalen Netzen ist oft verknüpft mit einer Verbesserung ihrer Leistung, da dadurch mehr Level von Abstraktion ermöglicht werden. Da LSTMs mit Sequenzen arbeiten, bedeutet das in diesem Fall, dass die Interpretation der Informationen im Zeitverlauf mehr Abstraktion erhält. Deshalb macht es Sinn, für herausfordernde Vorhersagen von Zeitsequenzen mehrere LSTM-Schichten aneinander zu reihen. (16)
Deshalb wurde zusätzlich eine Architektur mit zwei LSTM-Schichten und eine mit drei LSTM-Schichten getestet.
Auch die versteckten Schichten („hidden Layer“) des LSTM wurden dabei zwischen 128 und 256 variiert – zu wenige hidden Layer können die Leistung des Modells, komplexe Muster zu erlernen einschränken, zu viele können zu Overfitting, hohem Rechenaufwand und langsamerer Konvergenz führen.

Wie bereits dargestellt haben frühere Forschungsarbeiten gezeigt, dass die Kombination von LSTMs mit CNNs bei der Analyse von EKGs zielführend sein kann.
Es wurde die Kombination von einem LSTM mit einer LSTM-Schicht mit einem CNN mit drei Convolutional Schichten getestet.
Davon ausgehend gab es auch eine Architektur, die aus einer LSTM-Schicht mit 5 Convolutional Schichten bestand.
Schließlich wurden Modelle aus 2 LSTM-Schichten und 3 Convolutional Schichten getestet.
Bei den Convolutional Layers wurde bei Erhöhung der Anzahl der Layer auf fünf auch die kernel_size von (5,1) auf (10,1) geändert. Von der kernel_size hängt ab, wie groß die Bereiche sind, die ein Neuron betrachtet, wodurch die Fähigkeit des Netzes beeinflusst wird, verschiedene Detailinformationen zu erfassen. (17)

Zu Beginn der Studie wurde der Adam Optimizer verwendet und hierbei verschiedene Learning Rates (0,0001 und 0,00001) getestet. Adam ist ein „adaptive Optimizer“, was bedeutet, dass er größere Schritte bei der Minimierung der Loss Funktion ausführt, wenn sich die Gradienten nur wenig ändern und kleine Schritte, wenn sich die Gradienten stark ändern. (18)
Der Adam Optimizer ist gut geeignet für Probleme mit hohen Datenmengen, wie im Fall von EKG-Analysen, und sehr effektiv, weshalb er weit verbreitet ist.
Es wurde mit verschiedenen Learning Rates experimentiert – eine zu kleine Learning Rate könnte in einem sehr langsamen Training resultieren, eine zu große könnte bedeuten, dass bei der Minimierung der Loss Funktion das Minimum übersprungen wird, und somit keine Konvergenz, sondern Divergenz erreicht werden würde, bei dem die idealen Modellparameter, um das beste Ergebnis zu erzeugen, nicht gefunden werden würden. (19)

Weiterhin wurde der AdamW Optimizer getestet. Hierbei handelt es sich um eine Weiterentwicklung von Adam, in der der Weight Decay entkoppelt vom eigentlichen Optimierungsschritt stattfindet. Durch Weight Decay sollen Gewichte eher klein gehalten werden, was zu geringerem Overfitting und verbesserter Optimierung führt. (18) Es wurde mit verschiedenen Werten von Weight Decay experimentiert: 0,1, 0,01, 0,001.

Dazu passend wurden Versuche mit zwei verschiedenen Learning Rate Schedulern unternommen. Learning Rate Scheduler können dabei unterstützen, bei steigender Epochenzahl die Lernrate gering zu halten, um Divergenz bei der Suche nach dem Minimum der Loss Funktion zu vermeiden. (20)
Es wurde ein Learning Rate Scheduler getestet, der die Lernrate in jeder Epoche mit 0,95 multipliziert, der andere multipliziert die Lernrate in jeder 30. Epoche mit 0,1. Bei letzterem handelte es sich eher um ein Experiment aus Neugier.

Schließlich wurde ein Dropout von 0,5 in einigen Architekturen getestet.
Dropout wird verwendet, um Overfitting in neuronalen Netzwerken zu reduzieren, indem zufällig ausgewählte Neurone mit einer bestimmten Wahrscheinlichkeit (hier 0,5) während des Trainings ausgeschaltet werden. Dies führt dazu, dass das Netzwerk während des Trainings verschiedene neuronale Pfade durchläuft und somit eine breitere Vielfalt an Merkmalen lernt. Dadurch soll das Netz robuster werden und besser generalisieren. (21)

In manchen Architekturen wurde Batch Normalization verwendet. Batchnormalisierung wird eingesetzt, um das Training von neuronalen Netzwerken zu beschleunigen und zu stabilisieren, indem die Aktivierungen in jeder Schicht normalisiert werden. Dabei werden die Aktivierungen so skaliert, dass sie eine mittlere Aktivierung und eine Standardabweichung von eins haben, was dazu beiträgt, dass das Modell schneller konvergiert und weniger anfällig für Überanpassung wird. (22)

Jedes Netz wurde 1000 Epochen lang trainiert. Checkpoints wurden für die Epoche mit der höchsten Balanced Accuracy auf dem Validierungsdatensatz gespeichert. Der Split in Trainings- und Validierungsdatensatz betrug 0,8 zu 0,2. Wenn es zu einer Patientin/einem Patienten mehrere Messungen gab, waren diese entweder nur im Trainings- oder Validierungsdatensatz zu finden.

Ergebnisse

Verschiedene Kombinationen der eben vorgestellten Datensätze und Machine Learning Techniken wurden getestet.
Einen Überblick über alle Kombinationen und die resultierende Balanced Accuracy im Trainings- und Validierungsdatensatz bietet Tabelle 1. Herausstechende Ergebnisse sollen im Folgenden betrachtet werden.

Wenn nicht anders angegeben, ist bei „Accuracy“ im Folgenden immer von der Balanced Accuracy auf dem Validierungsdatensatz die Rede.

Zunächst fällt auf, dass eine Learning Rate (LR) von 0,00001 bei Adam Optimizer zu klein ist, um ein Netz praktikabel zu trainieren:

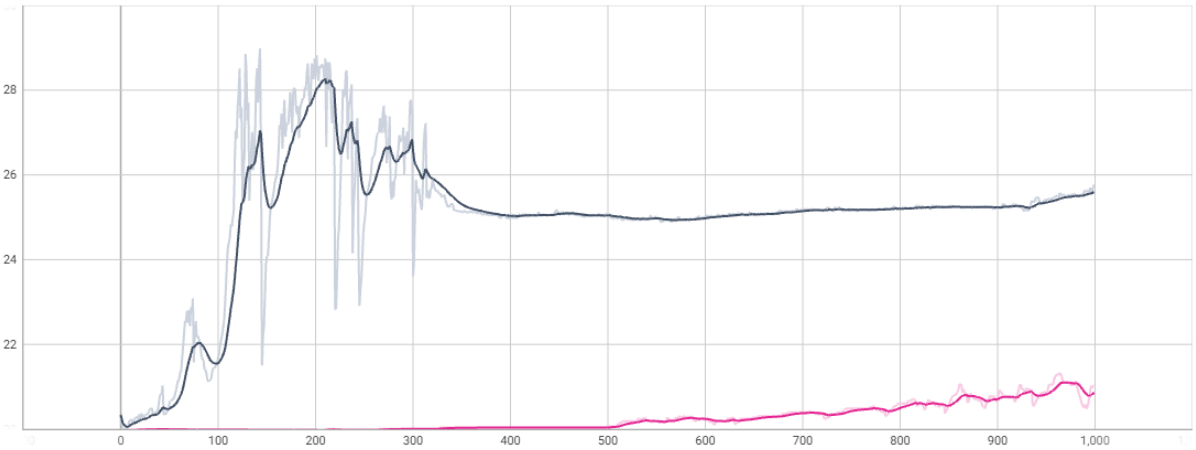

Abb. 4: Balanced Accuracy zweier identischer LSTM-Architekturen, der einzige Unterschied liegt in der Learning Rate: 0,00001 (pink) und 0,0001 (grau). x-Achse: Epochen, y-Achse: Balanced Accuracy.

Während das Modell mit einer Learning Rate von 0,0001 bei ca. 210 Epochen bereits ein Maximum der Balanced Accuracy auf dem Validierungsdatensatz erreicht hatte, begann die Accuracy des Modells mit einer LR von 0,00001 bei 1000 Epochen gerade erst langsam zu Steigen. Deshalb wurde mit einer Learning Rate von 0,0001 fortgefahren.

Weiterhin konnte festgestellt werden, dass tiefere LSTM-Architekturen zu einer verbesserten Leistung des Modells führen.

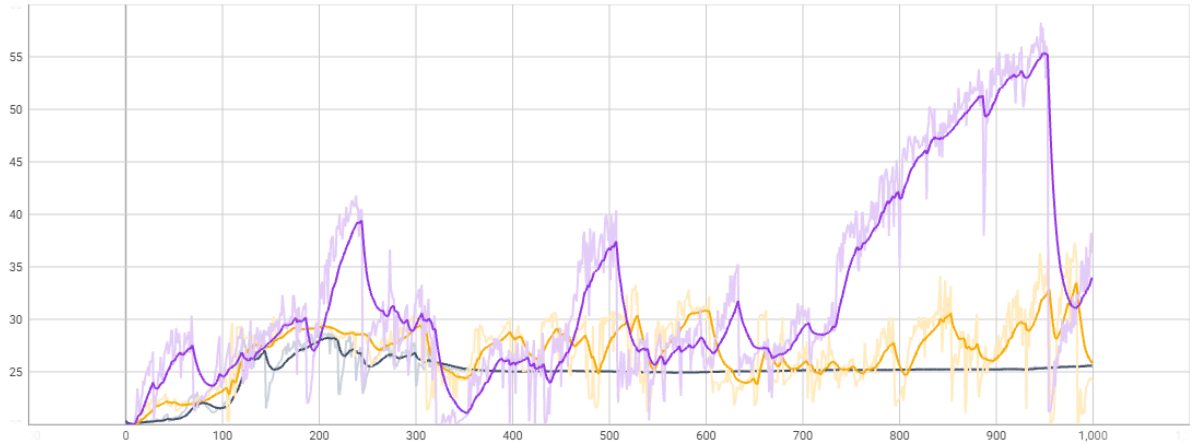

Abb. 5: Drei verschiedene LSTM-Architekturen: nur eine LSTM-Schicht (grau), zwei LSTM-Schichten (gelb) und drei LSTM-Schichten (lila). x-Achse: Epochen, y-Achse: Balanced Accuracy.

Der Graph zeigt die Balanced Accuracys im Validierungsdatensatz von einem Standard LSTM-Modell mit einer Schicht (grau), zwei Schichten (gelb) und drei Schichten (lila). Die Accuracy des LSTM mit drei Layern übersteigt mit 58,26% die der anderen beiden Netze (28,82% bei einer Layer, 37,27% bei zwei Layern) bei weitem.

Außerdem war die Kombination der LSTM- mit CNN-Schichten sehr vielversprechend.
In der folgenden Grafik ist in dunkelblau eine LSTM-Architektur mit einem „hidden layer“-Wert von 128 abgebildet, die mit drei Convolutional Schichten verbunden wurde. In grün wurde der „hidden layer“-Wert durch 256 ersetzt. Dieser Architektur wurde noch eine Schicht mit einer Dropout Wahrscheinlichkeit von 0,5 hinzugefügt, dies ist in orange abgebildet.
Wie aus der Grafik ersichtlich, ist die balancierte Accuracy dieser drei Modelle bei höherer Epochen Anzahl sehr ähnlich, bei genauerer Betrachtung (s. Tabelle 1) kann man argumentieren, dass das beste Modell dieser drei das mit einem „hidden layer“-Wert von 256 und Dropout ist. Dies ist auch nachvollziehbar – die tiefere Netzwerkarchitektur ermöglicht ein höheres Abstraktionslevel, während der Einsatz von Dropout Overfitting vermeidet.

In pink ist noch die Verbindung einer LSTM-Schicht mit fünf Convolutional Schichten abgebildet. Es ist sichtbar, dass dieses Modell zwar schneller eine höhere Accuracy erreicht, als die anderen, jedoch bei steigender Epochenanzahl unter den anderen zurückbleibt. Dies ist möglicherweise dadurch zu erklären, dass die erhöhte Anzahl an Convolutional Schichten ein zu hohes Abstraktionslevel bewirkt, wodurch einzelne Details des Trainingsdatensatzes erlernt werden, was Overfitting bedingen kann. Ein weiteres Indiz dafür ist, dass in dieser Architektur die balancierte Accuracy im Trainingsdatensatz 100% erreicht.

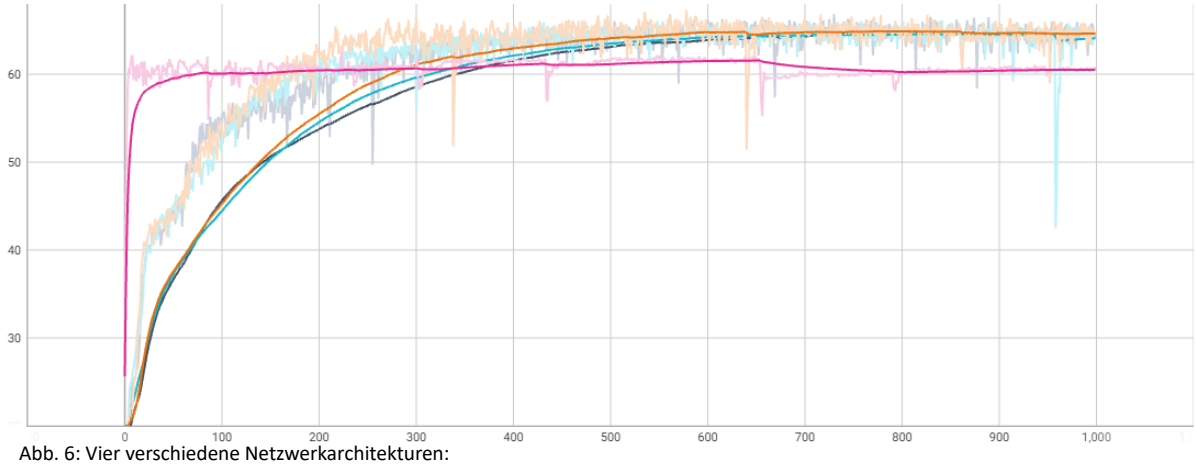

Abb. 6: Vier verschiedene Netzwerkarchitekturen:
- LSTM mit einer Schicht und hidden_layer=128 in Kombination mit drei Convolutional Schichten (dunkelblau)
- LSTM mit einer Schicht und hidden_layer = 256 in Kombination mit drei Convolutional Schichten (grün)
- LSTM mit einer Schicht, hidden_layer = 256 in Kombination mit drei Convolutional Schichten und dropout = 0,5 (orange)
- LSTM mit einer Schicht, hidden_layer = 256 in Kombination mit fünf Convolutional Schichten und dropout = 0,5 (pink)
x-Achse: Epochen, y-Achse: Balanced Accuracy.

Die Batchnormalization führte zwar zu einer früheren Konvergenz des Trainings, aber zu etwas schlechteren Accuracys, weshalb sie nach zwei Versuchen in Modellen aus LSTM- und Convolutional Schichten nicht weiterverwendet wurde:
Eine Architektur mit einer LSTM-Schicht und drei Convolutional Schichten und einer Schicht mit Dropoutwahrscheinlichkeit von 0,5 erreichte eine Accuracy von 67,35%. Mit Batchnormalization erreichte die gleiche Architektur 66,34%.

Die Verknüpfung von zwei LSTM-Schichten mit drei Convolutional Schichten wurde auf dem Standarddatensatz mit 1000 Messpunkten auf 12 Kanälen getestet, außerdem auch auf dem Datensatz mit randomisiertem Start und dem, der nur die letzten 200 Messpunkte jedes Kanals betrachtet.

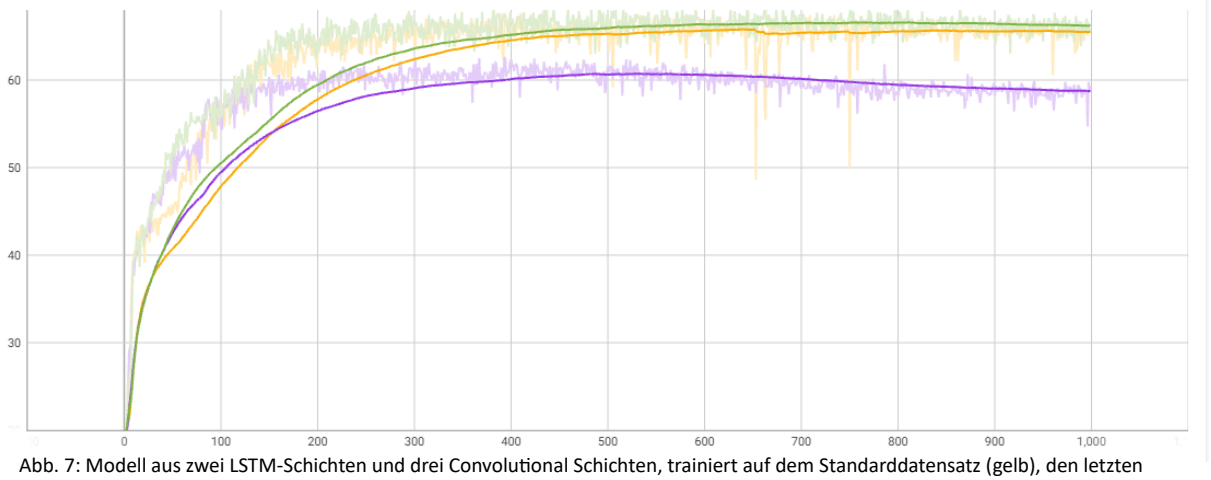

Abb. 7: Modell aus zwei LSTM-Schichten und drei Convolutional Schichten, trainiert auf dem Standarddatensatz (gelb), den letzten 200 Messpunkten des Standarddatensatzes (lila) und dem Datensatz mit randomisiertem Start (grün).
x-Achse: Epochen, y-Achse: Balanced Accuracy.

Das Training mit dem Datensatz aus den letzten 200 Messpunkten (lila) hat die schlechteste Balanced Accuracy, woraus sich interpretieren lässt, dass das LSTM längere Sequenzen als 200 Messpunkte gut verarbeiten kann. In grün ist der Datensatz mit dem zufälligen Startpunkt abgebildet, in gelb der Standarddatensatz. Es ist erkennbar, dass der randomisierte Start die Ergebnisse des Netzes etwas besser macht, wahrscheinlich weil es durch diese Art der Datenaugmentation robuster wird.

Auch wurde deutlich, dass das Netz, dass aus zwei LSTM- und drei Convolutional Schichten (gelb) besteht eine bessere Leistung hat, als ein Netz, dass aus einer LSTM- und drei Convolutional Schichten (orange) besteht:

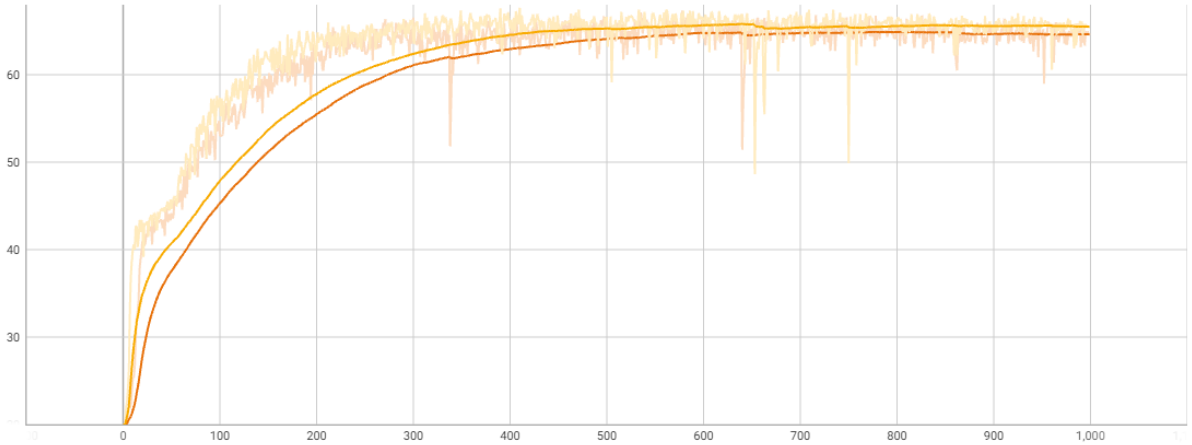

Abb. 8: Ein Modell aus zwei LSTM-Schichten und drei Convolutional Schichten (gelb) und ein Modell aus einer LSTM-Schicht und drei Convolutional Schichten (orange).
x-Achse: Epochen, y-Achse: Balanced Accuracy.

Aufgrund dieser Ergebnisse wurden die weiteren Experimente mit einer Architektur aus zwei LSTM- und drei Convolutional Schichten auf dem Datensatz mit randomisiertem Start durchgeführt.

Die Experimente mit dem Learning Rate Scheduler erwiesen sich als nicht zielführend. Die Multiplikation der Learning Rate mit 0,95 in jeder Epoche führte auf 1000 Epochen zu einer balanced Accuracy von 44,54%, die Multiplikation mit 0,1 alle 30 Epochen zu einer balanced Accuracy von 45.8%.

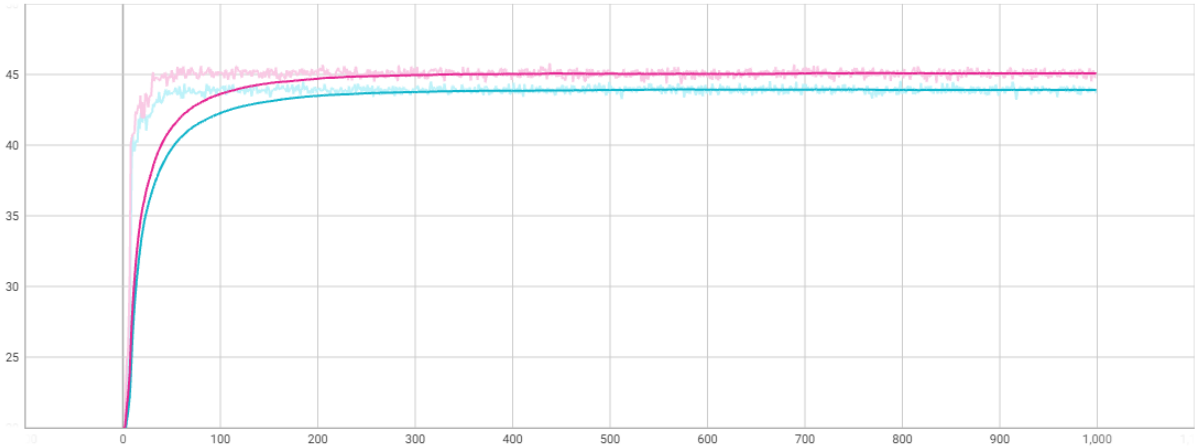

Abb. 9: Ein Modell aus zwei LSTM-Schichten und drei Convolutional, einmal mit einem LR-Scheduler, der die LR in jeder Epoche mit 0,95 multipliziert (türkis) und einmal mit einem LR-Scheduler, der die LR jede 30. Epoche mit 0,1 multipliziert.
x-Achse: Epochen, y-Achse: Balanced Accuracy.

Auch war nach ca. 250 Epochen noch kaum ein Anstieg der Balanced Accuracy erkennbar. Eine mögliche Erklärung hierfür ist, dass durch den Einsatz des LR Scheduler die LR so klein wird, dass sich an den Gewichten des Netzes nichts mehr verändert und dadurch auch die Accuracy gleichbleibt.

Zuletzt wurde in dem Modell, das aus zwei LSTM- und drei Convolutional Schichten mit Dropout bestand, noch der Adam Optimizer durch den AdamW Optimizer ausgetauscht.

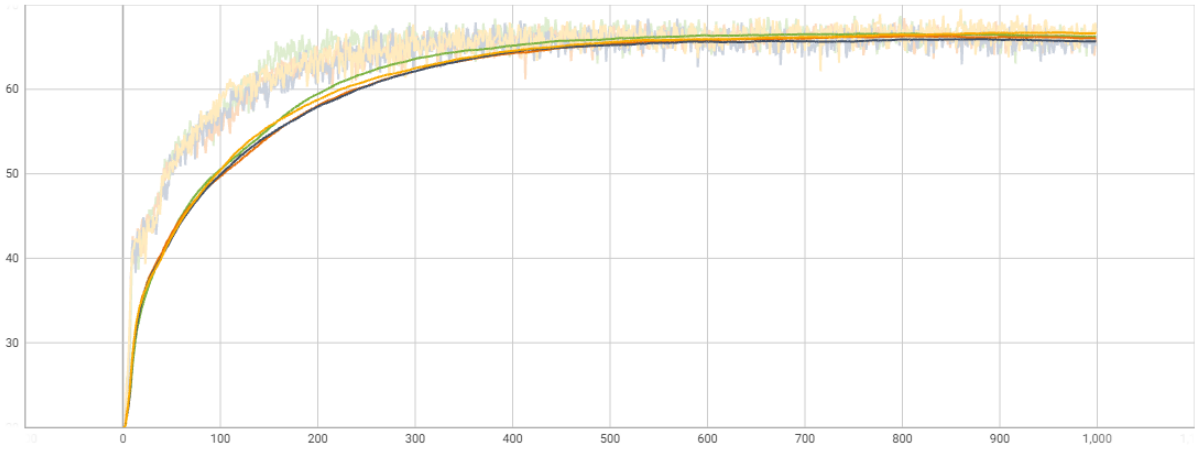

Abb. 10: Modell aus zwei LSTM-, einer Convolutional und einer Dropout Schicht. Einziger Unterschied ist die Variation im Optimizer:
- Adam, LR = 0,0001 (grün),
- AdamW, LR = 0,0001, weight_decay = 0,001 (orange),
- AdamW, LR = 0,0001, weight_decay = 0,01 (grau),
- AdamW, LR = 0,0001, weight_decay = 0,1 (gelb).
x-Achse: Epochen, y-Achse: Balanced Accuracy.

In den Graphen ist zunächst kein großer Unterschied in der Leistung von Adam (grün) und AdamW mit einem Weight Decay von 0,001 (orange), 0,01 (grau) und 0,1 (gelb) zu erkennen. Die von Adam erreichte Accuracy wirkt sogar etwas besser, als die von AdamW.
Bei Betrachtung der Tabelle stellt man fest, dass AdamW mit 69,51% knapp das beste Ergebnis erreicht, wobei sich die Accuracys aller vier Variationen zwischen 68,38% und 69,51% bewegen und der Unterschied zwischen ihnen somit nicht nennenswert ist. Mit zunehmender Komplexität von Modellen steigt auch der Benefit, den ihr Training durch AdamW hat. Möglicherweise sind die vorliegenden Architekturen zu einfach, um sichtbar von der Verwendung von AdamW zu profitieren.

Die folgende Tabelle bietet eine Übersicht über alle getesteten Kombinationen aus Datensätzen und Machine Learning Techniken. Aus der durchgeführten Studie geht hervor, dass tiefere Modell-Architekturen bessere Ergebnisse erzählen. Ganz knapp hatte das Modell die beste Accuracy, das aus zwei LSTM- Schichten (mit 256 Hidden Layers) in Kombination mit drei Convolutional und einer Dropout Schicht mit Wahrscheinlichkeit von 0,5 besteht. Weiterhin ist dieses Modell mit einer Learning Rate von 0,0001 und einem Weight Decay von 0,1 bei Einsatz von AdamW konfiguriert. Trainiert wurde dieses Modell auf dem Datensatz mit randomisiertem Start.

Netzwerk	No. LSTM Layer	No. Hidden Layer (LSTM)	Conv Layers	Optimizer	LR	Weight Decay	Dropout	LRStep	Datensatz	Batchnorm	Train Acc.	Val Acc.
LSTM	1	128	-	Adam	0,0001	-	-	-	Standard	-	29,4	28,82
LSTM	1	128	-	Adam	0,00001	-	-	-	Standard	-	21,72	21,33
LSTM	2	128	-	Adam	0,0001	-	-	-	Standard	-	37,36	37,27
LSTM	3	128	-	Adam	0,0001	-	-	-	Standard	-	59,84	58,26
LSTM_Conv	1	128	3	Adam	0,0001	-	-	-	Standard	-	80,27	66,86
LSTM_Conv	1	128	3	Adam	0,0001	-	0,5	-	Standard	-	79,7	66,31
LSTM_Conv	1	256	3	Adam	0,0001	-	-	-	Standard	-	82,38	66,95
LSTM_Conv	1	256	3	Adam	0,0001	-	0,5	-	Standard	-	81,1	67,35
LSTM_Conv	1	256	3	Adam	0,0001	-	0,5	-	Standard	yes	99,99	66,34
LSTM_Conv	1	256	5	Adam	0,0001	-	0,5	-	Standard	yes	100	62,65
LSTM_Conv	2	256	3	Adam	0,0001	-	0,5	-	Standard	-	85,69	67,66
LSTM_Conv	2	256	3	Adam	0,0001	-	0,5	-	letzte 200 Datenpunkte	-	95,82	62,56
LSTM_Conv	2	256	3	Adam	0,0001	-	0,5	-	Random Start	-	80,8	68,69
LSTM_Conv	2	256	3	Adam	0,0001	-	0,5	0,95	Random Start	-	44,78	44,54
LSTM_Conv	2	256	3	Adam	0,0001	-	0,5	30	Random Start	-	45,02	45,8
LSTM_Conv	2	256	3	AdamW	0,0001	0,1	0,5	-	Random Start	-	79,48	69,51
LSTM_Conv	2	256	3	AdamW	0,0001	0,01	0,5	-	Random Start	-	80,67	68,38
LSTM_Conv	2	256	3	AdamW	0,0001	0,001	0,5	-	Random Start	-	79,6	68,43

Tab. 1: Übersicht über die verschiedenen Modellkonfigurationen, die in dieser Studie getestet wurden

Verwendung des Codes

Zunächst soll das Repository geklont werden. Weiterhin muss der PTB-XL Datensatz von PhysioNet heruntergeladen werden.

Die Python Skripte des Repositorys müssen modifiziert werden:

In import_data.py muss daraufhin der Pfad für den Import der Trainingsdaten angepasst werden, hier wird der Pfad eingefügt, der auf den Ordner zeigt, in dem sich die entsprechenden Ordner für die 100Hz und 500Hz Aufnahmen befinden (s. Codebeispiel Zeile 16-18).

Zum Trainieren eines Netzwerks muss nur die main.py ausgeführt werden. In der Version, in der der Code aus dem Repository heruntergeladen wird, wird die beste getestete Netzwerkarchitektur trainiert, also zwei LSTM- mit drei Convolutional Schichten, Dropout und AdamW mit weight_decay=0,1 auf dem randomisierten Datensatz.

Alternativ können verschiedene Änderungen des Codes durchgeführt werden, um das Modell auf die gewünschte Weise zu verändern:
In Zeile 36 von main.py kann die „net“-Variable gleichgesetzt werden mit der Netzarchitektur, die man testen möchte. Zur Auswahl stehen „LSTM“ (eine LSTM-Schicht), „LSTM_2stacked“ (zwei LSTM-Schichten), „LSTM_3stacked“ (drei LSTM-Schichten), „LSTM_3Conv“ (zwei LSTM- und drei Convolutional Schichten) und „LSTM_5Conv“ (eine LSTM- und fünf Convolutional Schichten).

Weiterhin kann in Zeile 26 und 29 der Datensatz ausgetauscht werden, zur Verfügung stehen „ECGDatasetUpdate“ (Standarddatensatz), „ECGDataset200“ (letzte 200 Datenpunkte) und „ECGDatasetRandomStart“.

Außerdem können durch Variationen der Zeilen 14-16 (Adam) bzw. 18-20 (AdamW) in der Datei trainloop.py verschiedene Variationen der Optimizer, ihrer Learning Rates und des Weight Decays getestet werden.
In der gleichen Datei befindet sich der Learning Rate Scheduler. In Zeile 22-24 kann die Version getestet werden, die in jeder Epoche die LR modifiziert, in Zeile 26-27 die Version, bei der durch eine step_size festgelegt werden kann, in jeder wievielten Epoche die LR verändert werden soll. Um den LR Scheduler im Code tatsächlich einzusetzen, muss Zeile 61 in main.py außerdem einkommentiert werden.

Durch Veränderung der Werte „hidden_size“ und „num_layers“ in der LSTM-Schicht aller Architekturen kann das Netz außerdem tiefer oder weniger tief gestaltet werden (z.B. in Zeile 19 von lstm_3conv.py).

Batchnormalization kann hinter jeder Convolutional Schicht ein- und auskommentiert werden.

Dropout kann im forward pass jeder Architektur ein- oder auskommentiert werden (z.B. in Zeile 35 von lstm_3conv.py).

Weiterhin sind über die readme auf github verschiedene Checkpoints zur Verfügung gestellt, welche mit der gdown Bibliothek einfach als URL in den Code geladen werden können.
Das Netz kann an dem Punkt mit der auf 1000 Epochen besten Balanced Accuracy (im Validierungsdatensatz) weiter trainiert werden.
Hierfür muss der entsprechende Link (in der readme) für die url-Variable in Zeile 23 von resume_training_fromcheckpoint.py eingefügt werden. Nachdem das Netz entsprechend den oben gemachten Erklärungen für die jeweilige Architektur des Checkpoints angepasst worden ist, kann resume_training_fromcheckpoint.py ausgeführt werden, um das Training fortzusetzen.

Fazit & Ausblick

In der vorliegenden Studie wurden verschiedene Kombinationen von Machine Learning Techniken und Datensätzen getestet, um das beste Modell für die Interpretation von EKG-Daten zu finden. Konsistent mit der Literatur erwies sich dabei eine Kombination aus LSTM- und Convolutional Schichten am zielführendsten. Die hohen Accuracy-Werte von 99,95 von Oh, Tan und Kollegen (6) (7) konnten hierbei leider nicht erreicht werden.

Es gibt verschiedene Möglichkeiten, die ausprobiert werden können, um die Leistung des Netzes zu verbessern.
So könnten Modelle mit tieferen LSTM-Architekturen gewählt werden, um die Performance zu verbessern. Die Erhöhung der LSTM-Schichten von eins auf zwei und dann auf drei hatte vielversprechende Ergebnisse gezeigt, genauso die Steigerung der hidden Layers des LSTM auf 256.

In Verbindung mit den Convolutional Schichten brachte die Erhöhung von einer LSTM-Schicht auf zwei LSTM-Schichten nur eine geringe Verbesserung der Balanced Accuracy im Validierungsdatensatz (unter ein Prozent). Deshalb wurde eine Kombination mit drei LSTM-Schichten und Convolutional Networks nicht mehr getestet. Diese Möglichkeit soll in näherer Zukunft noch exploriert werden. Somit kann getestet werden, wie das Netzwerk durch weitere Steigerung der Schichtenanzahl noch verbessert werden kann, bevor es zu Overfitting kommt.

Eine weitere Möglichkeit, mit der diese Studie zeitnah weitergeführt werden soll, ist der Einsatz einer Transformer Architektur.
Auch Transformer werden zur Sequenzanalyse eingesetzt. Hier wird „Self-Attention“ verwendet, durch welche die Beziehung eines Teils der Sequenz (z.B. ein Wort eines Satzes) zu anderen Teilen der gleichen Sequenz (andere Worte) herausgefunden werden soll. Die einzelnen Sequenzteile werden sozusagen im Kontext zum Rest der Sequenz betrachtet. Dabei hilft „Self-Attention“, Sequenzanteile zu finden, die für die Interpretation eines bestimmten anderen Sequenzteils Relevanz haben können. Hierfür wird die ganze Sequenz auf einmal prozessiert, dadurch können Kontexte besser verstanden werden. Probleme von Kurz- und Langzeitgedächtnis, die durch LSTMs teilweise gelöst werden, sind somit nicht mehr vorhanden, da, wenn man eine ganze Sequenz betrachtet, einzelne Abhängigkeiten nicht mehr vergessen werden können.
Weiterhin berechnen Transformer bidirektional, was bedeutet, dass bei der Betrachtung bestimmter Sequenzanteile auch immer die Anteile davor und danach miteinbezogen werden. (23) (24)
Dieser Mechanismus kann für die Analyse von EKGs zielführend sein, da es sich auch hier um lange Sequenzen handelt, bei denen Abhängigkeiten innerhalb einzelner Sequenzen bestehen. Beispielsweise verwendeten bereits Hu und Kollegen diese Technologie, um Diagnosen aus EKG-Sequenzen mit einer Accuracy von über 99% bestimmen zu können. (25) Deshalb soll diese Technologie zeitnah in die vorliegende Studie integriert werden.

Anhang: Literaturverzeichnis

1. ***Bundesministerium für Soziales, Gesundheit, Pflege und Konsumentenschutz.***
https://www.sozialministerium.at/Themen/Gesundheit/Nicht-uebertragbare-Krankheiten/Herz-Kreislauf-Krankheiten.html, Aufruf, 17.02.2025.

2. ***doccheck.*** *https://flexikon.doccheck.com/de/Elektrokardiogramm, Aufruf, 17.02.2024.*

3. ***Selles M., Marina-Breysse M.*** *Current and Future Use of Artificial Intelligence in Electrocardiography. J Cardiovasc Dev Dis. 2023, 10(4), S. 175.*

4. ***Zhang J., Lui A., Gao M., Chen X., Zhang X., Chen X.*** *ECG-based multi-class arrhythmia detection using spatio-temporal attention-based convolutional recurrent neural network. 2020, Volume 106.*

5. ***Chang K., Hsieh P., Wu M., Wang Y., Chen J., Tsai F., Shih E., Hwang M., Huang T.*** *Usefulness of Machine Learning-Based Detection and Classification of Cardiac Arrhythmias With 12-Lead Electrocardiograms. 2021, Volume 37, S. 94-104.*

6. ***Oh S., Ng E., Tan R., Acharya U.*** *Automated diagnosis of arrhythmia using combination of CNN and LSTM techniques with variable length heart beats. Comput Biol Med. 2018, 102:278-287.*

7. ***Tan J., Hagiwara Y., Pang W., Lim I., Oh S., Adam M., Tan R., Chen M., Acharya U.*** *Application of stacked convolutional and long short-term memory network for accurate identification of CAD ECG signals. Computers in Biology and Medicine. 2018, Volume 94.*

8. ***Oh S., Adam M., Tan J., Hagiwara Y., Sudarshan V., Joh J., Chua K., Chua K.*** *Automated Identification Of Coronar Artery Disease From Short-Term 12 Lead Electrocardiogramm Signals By Using Wavelet Package Decomposition And Common Spatial Pattern Techniques. Journal of Mechanics in Medicine and Biology. 2017, Vol 17.*

9. ***Turnip A., Rizqywan M., Kusumandari D., Turnip M., Sihombing P.*** *Classification of ECG signal with Support Vector Machine Method for Arrhythmia Detection. Journal of Physics: Conference Series. 2018, 970.*

10. ***Rabee A., Barhumi I.*** *ECG signal classification using support vector machine based on wavelet multiresolution analysis. IEEE Concerences. 2012.*

11. databasecamp. ***https://databasecamp.de/ki/lstm.*** *Aufruf: 17.02.2024.*

*12. —.****https://databasecamp.de/en/ml/convolutional-neural-networks, Aufruf 17.02.2024.***

13. ***Acharya U., Fujita H., Oh S., Hagiwara Y., Tan J., Adam M.*** *Application of deep convolutional neural network for automated detection of myocardial infarction using ECG signals. Information Sciences. 2017, Volumes 415–416.*

14. ***Wagner P., Strodthoff N., Bousseljot R., Samek W., Schaeffter T.*** *PTB-XL, a large publicly available electrocardiography dataset. PhysioNet. 2022.*

15. ***Wagner P., Strodthoff N., Bousseljot R., Kreiseler D., Lunze F., Samek W., Schaeffter T.*** *PTB-XL, a large publicly available electrocardiography dataset. nature - Scientific Data. 2020, 154(7).*

16. Machine Learning Mastery. ***https://machinelearningmastery.com/stacked-long-short-term-memory-networks/, Aufruf 18.02.2024.***

17. Pandey S. ***https://medium.com/analytics-vidhya/how-to-choose-the-size-of-the-convolution-filter-or-kernel-size-for-cnn-86a55a1e2d15, Aufruf 18.02.2024.***

18. Graetz F. ***https://towardsdatascience.com/why-adamw-matters-736223f31b5d, Aufruf 18.02.2024.***

19. Machine Learning Mastery. ***https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/, Aufruf 18.02.2024.***

20. Sudhakar S.,. ***https://towardsdatascience.com/learning-rate-scheduler-d8a55747dd90, Aufruf 18.02.2024.***

21. Yadav, H. ***https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9, Aufruf 18.02.2024.***

22. Doshi, K. ***https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739, Aufruf 18.02.2024.***

23. ***Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A., Kaiser L., Polosukhin I.*** *Attention Is All You Need. arxiv. 2017.*

24. databasecamp. ***https://databasecamp.de/ki-blog/transformer-betreten-die-buehne, Aufruf 18.02.2024.***

25. ***Hu R., Chen J., Zhou L.*** *A transformer-based deep neural network for arrhythmia detection using continuous ECG signals. Computers in Biology and Medicine. 2022, Volume 144.*

Abbildungsquellen:

Abb.1: https://flexikon.doccheck.com/de/Sinusknoten

Abb.2: https://kardiologie-lippstadt.de/ekg/