Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

Bachelorarbeit (IT19ta_WIN)

Auswertung der Ausnutzung von zeitabhängigen segment-übergreifenden Merkmalen durch ein Vision Transformer-Modell

Autoren	**Safuan Achargui** **Chloé Nitschmann**
Hauptbetreuung	**Thilo Stadelmann**
Datum	**09.06.2023**

Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbstständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

Ort, Datum:	**Name Studierende:**
Winterthur, 09.06.23	Archargui, Safuan
Pfungen, 09.06.23	Nitschmann, Chloé

1 Inhaltsverzeichnis

2 Abstract

Im Umfang dieser Arbeit wurde untersucht, wie sich das Entfernen von zeitabhängigen segment-übergreifenden (SST) Merkmalen auf einen Transformer unter Verwendung von Spektrogrammen auswirkt. Aus dem Bereich der NLP ist bekannt [1], dass Transformer in der Lage sind, komplexe Zusammenhänge über lange Sequenzen zu modellieren.

SST-Merkmale, auch Sprachprosodie genannt, beschreiben Informationen wie Akzente oder zeitliche Abfolge von Lauten [2]. Es wird untersucht, ob und in welchem Ausmass Transformer SST-Merkmale verstehen. Neururer [3] hat einen Test entwickelt und angewendet, mit welchem die Ausnutzung dieser Eigenschaften gemessen werden kann.

Um eine Auswertung der Verwendung von SST-Merkmalen durchzuführen, wurde der Vision Transformer (VIT) von TIMM [4] verwendet. Darauf basierend sind unterschiedliche Trainingsstrategien implementiert und angewendet worden [5, 6]. Die gewonnen Erkenntnisse zeigen, dass unser VIT die SST-Merkmale nicht in signifikanter Weise beachtet. Dennoch kann dadurch abgeleitet werden, auf welchem Niveau SST-Merkmale durch den VIT verstanden werden. Es wurde gezeigt, dass der Vision Transformer globale Merkmale teilweise berücksichtigen.

3 Einführung

3.1 Motivation

Sprecherverifikation (SV), -identifikation (SI) und -bündelung (SC) sind Teilprobleme der Sprechererkennung (SR). Generell ist bei diesen das Ziel, anhand von Sprachaussagen den Sprecher zu erkennen. Neuronale Netzwerke sind effektiv in der Sprechererkennung [7, 8, 9]. Der entwickelte und angewendete Test von Neururer et al. [3] zeigt, dass bisherige neuronale Netzwerke hauptsächlich rahmenbasierte akustische (FBA) Eigenschaften modellieren. Bessere Resultate bei der SR werden erwartet, wenn nebst den FBA-Merkmalen auch SST-Merkmale modelliert werden [3, 10].

ChatGPT hat mit seinem Erfolg einen Trend in der Erforschung von Transformer losgetreten. Es werden nahezu wöchentlich neue Erkenntnisse oder Modelle in unterschiedlichen Teilgebieten der KI veröffentlicht und hierbei übertreffen diese vorhergehende Neuronale Netzwerke oftmals. Richtig trainiert, können Transformer lange Sequenzen modellieren und somit kontextabhängige Merkmale erkennen. Im Rahmen dieser Arbeit soll ein Transformer dazu genutzt werden SST-Merkmale kontextabhängig zu modellieren.

Transformer weisen auch Nachteile auf. Einer dieser Nachteile sind die grossen Mengen an Daten, welche benötigt werden, um Modelle zu trainieren. Ein vortrainiertes Modell kann dabei helfen diese Problematik zu reduzieren. Hierbei werden die Erfahrungen eines bestehenden Modells verwendet und diese durch das Finetuning auf kleinere Menge an Daten optimiert. Andernfalls kann ein eigenes Modell durch einen oftmals ressourcen- und zeitaufwendigen Trainingsprozess erarbeitet werden. Wir haben uns für Letzteres entschieden und wenden die MAE-Trainingsmethodik [5] in leicht abgeändert Form an.

3.2 Problemstellung

Neururer [10] stellt eine Methode vor, um Segmente von SST-Merkmalen zu befreien. Bei dieser wird zu Beginn ein Spektrogramm aus den Audiodaten generiert. Aus diesem wird ein Segment ausgeschnitten, welches als Originalsegment (OS) bezeichnet wird. Es wird als Basiswert verwendet. Um den ersten Vergleichswert zu erhalten, wird ein Aussagenmischsegment (SU) erzeugt. Dieses mischt die Frames und schneidet anschliessend ein Segment aus. Für einen weiteren Vergleich wird ein Mischsegment (SS) erzeugt. Hier wird zuerst das Segment ausgeschnitten und darauffolgend werden die Frames durchmischt. Damit wird die Nutzbarkeit von SST-Merkmalen reduziert und eine Prüfung der Nutzung durch Modelle, welche sich nur auf rahmenbasierte Akustik (FBA) verlassen, wird möglich [3]. Die Resultate sprechen dagegen, dass die durch von Neururer [10] überprüften Modelle SST-Merkmale zu einem nützlichen Grad modellieren.

Ein Vergleich zwischen OS und SS wird in dieser Arbeit dazu verwendet, um auf einem bisher ungetesteten Transformer die SST-Ausnutzung zu eruieren. Spezifisch wurde hierbei ein Vision Transformer (VIT) überprüft. Es wird angenommen, dass Transformer prädestiniert dazu sind, zeitabhängige segment-übergreifende Merkmalen auszunutzen.

3.3 Implementation

Das in dieser Arbeit entwickelte Modell, genutzte Skripte zur Erstellung von Grafiken, Modelle sind unter https://github.com/SafuanA/VisionTransformer verfügbar.

4 Theoretische Grundlagen

Die theoretischen Grundlagen in Bezug auf Sprecherverifikation (SV) werden in diesem Kapitel genauer erläutert. Hierbei liegt der Fokus auf Audiosignalverarbeitung und Transformern.

4.1 Mel-Spektrogramm

Mel-Spektrogramme sind eine grafische Darstellung von Audioinformationen. Auf der X-Achse wird die Zeit und auf der Y-Achse die Frequenz dargestellt. Durch einen farbigen Grafen wird die Amplitude zu einem spezifischen Zeitpunkt abgebildet. Bei vertieftem Verständnis der Mel-Spektrogrammen können neben Vokalen auch die Sprecher abgelesen werden [11]. Neurale Netzwerke sind in der Lage relevante Merkmale aus den Mel-Spektrogrammen auszulesen.

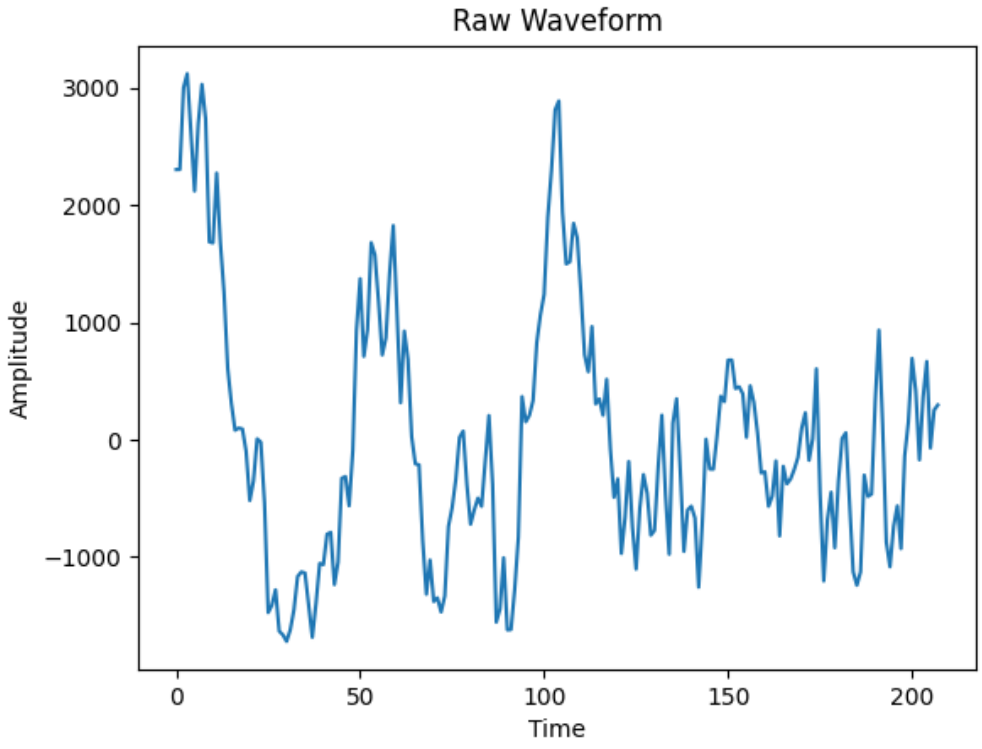


Abb 1 Rohsignalausschnitt einer VoxCeleb1 Audiodatei

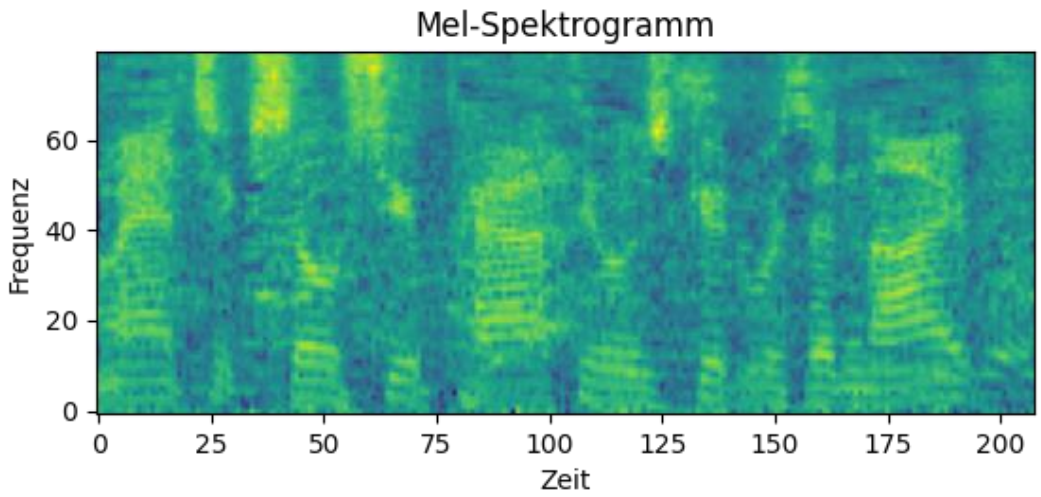


Abb 2 Mel-Spektrogramm des vorhergehenden Rohsignals

Bevor Signale verarbeitet werden, wird häufig eine Pre-Emphasis angewendet. Diese dient dazu die Amplitude eines Signales x zum Zeitpunkt t anhand des Signals zum Zeitpunkt t - 1 zu pitchen. Hierdurch werden hochfrequente Komponenten verstärkt, Signal-to-Noise-Rate reduziert, sowie numerischen Problemen bei der FFT ausgewichen. α ist hierbei zwischen 0.95 und 1 angesiedelt. [12]

$$preemphasis(t) = x(t) - \alpha * x(t-1)$$

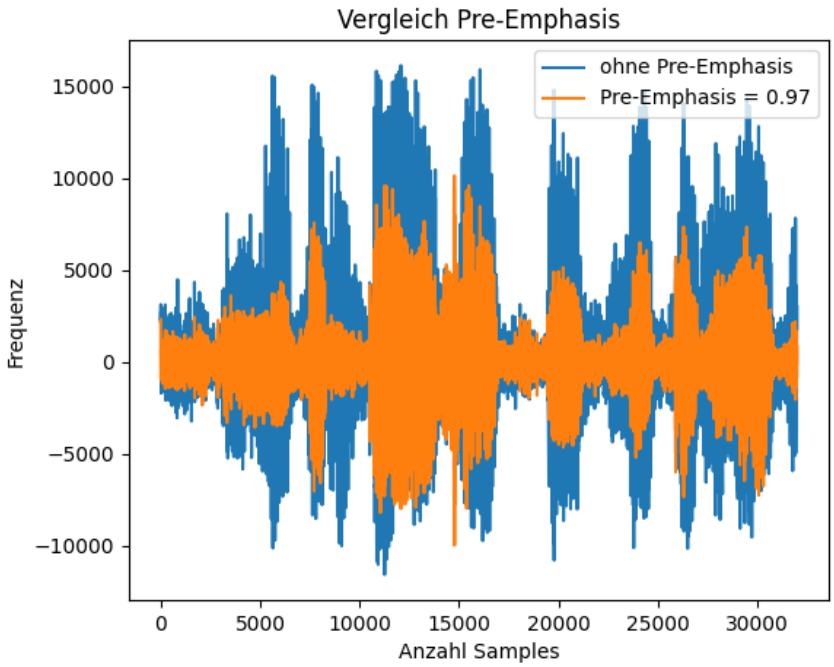


Abb 3 Vergleich eines Signals ohne Pre-Emphasis Koeffizienten respektive mit einer Pre-Emphasis von 0.97

Nach dem Fourier Theorem kann ein zeitlicher Signalverlauf durch eine Komposition von Cosinus- und Sinusschwingungen abgebildet werden.

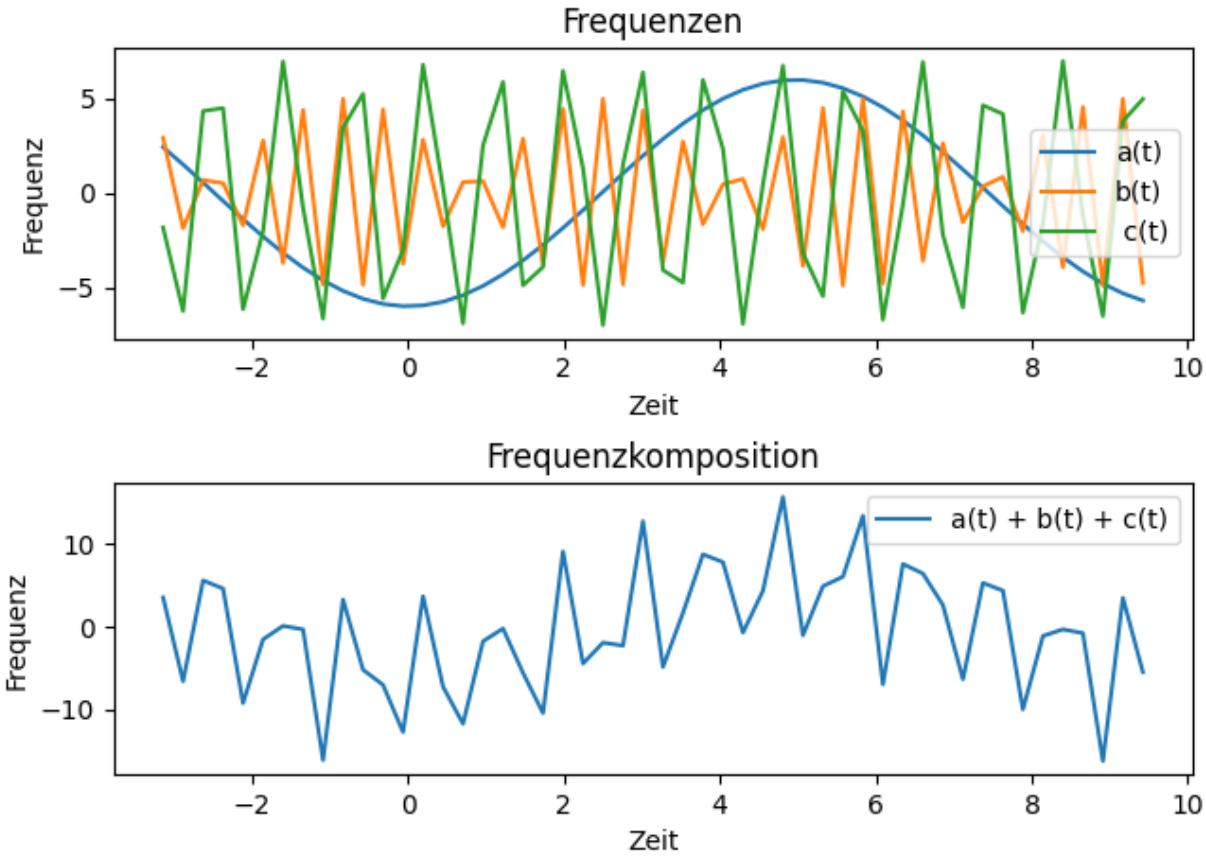


Abb 4 oben: Mehrere Signalfrequenzen,
unten: Komposition aus den Signalen der oberen Abbildung

Um Mel-Spektrogramme zu erhalten, werden Informationen zum Frequenzbereich eines Signales benötigt. Diese können durch eine Fourier-Transformation erhalten werden. [13]

Durch diese Transformation wird jede Frequenz in Relation zu derer Amplitude gestellt. Im Allgemeinen wird aufgrund der Komplexität der Diskreten Fourier-Transformation von $O(n^2)$, auf die Schnelle Fourier-Transformation (FFT) zurückgegriffen. Diese reduziert die Komplexität auf O(n*log(n)). Um dies zu erreichen, wird das Audiosignal in kürzere Abschnitte unterteilt, welche auch als Fenster bezeichnet werden. [13]

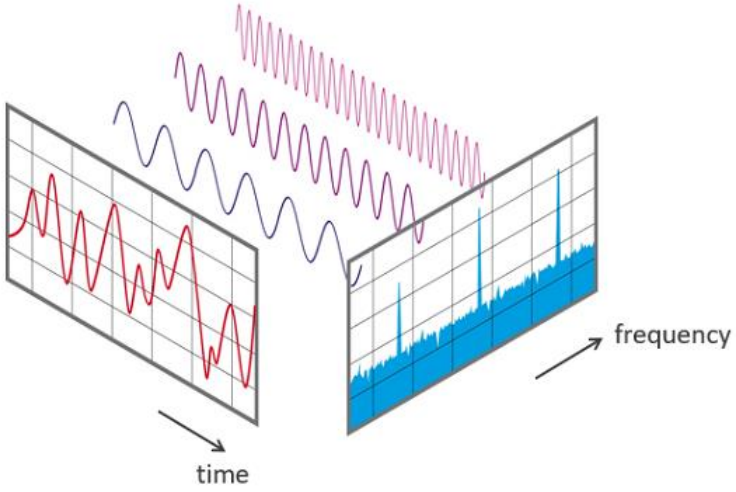


Abb 5 Visualisierung einer Fourier-Transformation entnommen aus [14]

Für die Kalkulation der Mel-Spektrogramme wird eine zeitdiskrete Kurzzeit-Fourier-Transformation mit Hamming-Fenster verwendet. Das Signal wird hierbei als Signalfolge von k bis Fensterbreite M-1 abgebildet. Hop-Länge h beschreibt die Distanz ohne Überlappung. Das kalkulierte Zeitfenster ist durch m gegeben. w stellt das Frequenzband dar. j ist $\sqrt{-1}$. Input stellt das Eingangssignal zu einem Zeitpunkt dar. Die Funktion $input(mhk)$ ist somit das Signal an der k-ten Position im m-ten Zeitfenster unter Berücksichtigung der Überlappung.

$$STFT(w,\ m)\ =\ \sum_{k=0}^{M-1} window(k) * input(m * h + k) * \exp\left(-j\frac{2\pi * w * k}{M}\right)$$

Die Hamming-Fensterfunktion $window(t)$ ist gegeben als

$$window(t)\ =\ \alpha\ -\ \left(\beta * \cos\left(\frac{2\pi * t}{M-1}\right)\right)$$

wobei α und β zusammen 1 ergeben.

Auf die resultierende Matrix wird eine Bandfilterung angewendet. Hierbei kommen Mel-Filterbänke zum Einsatz, welche die Daten, ähnlich der Wahrnehmung des menschlichen Gehörs, kategorisieren. Mel-Filterbänke sind eine Menge von dreieckigen Filtern, welche anhand der Mel-Skalierung verteilt sind. Das Anwenden von Mel-Filterbänken sorgt dafür, dass die Frequenzen im Mel-Spektrogramm logarithmisch dargestellt werden. [15]

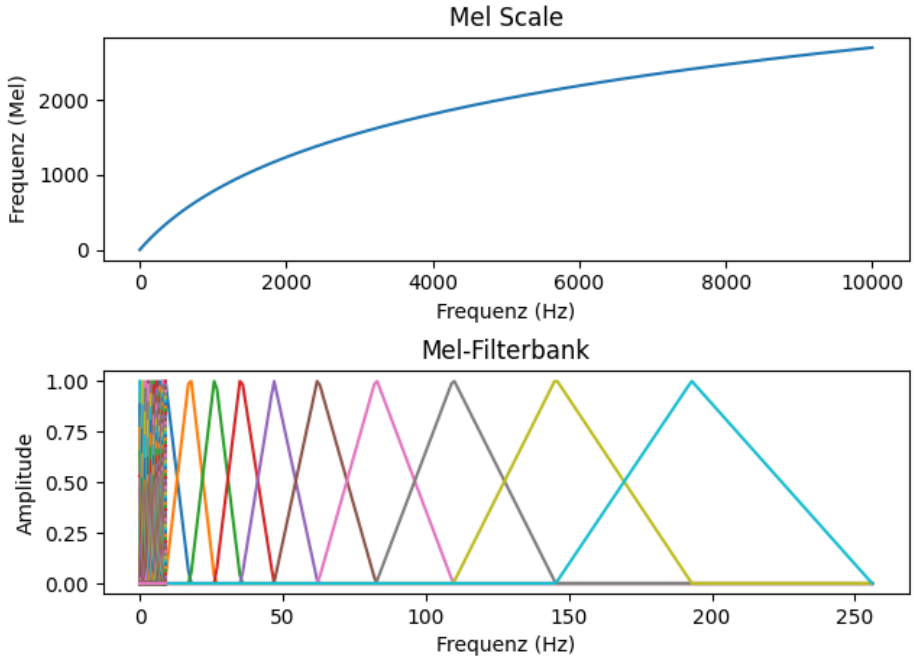


Abb 6 Mel-Skalierung und Mel-Filterbänke mit n_mels = 10

4.2 Sprachprosodie

Sprachprosodie oder auch zeitabhängige segment-übergreifende Merkmale (SST), bezeichnen spezifische Informationen, welche innerhalb eines Audiosignals enthalten sind, wobei in dieser Arbeit ein Fokus auf Sprachaussagen von Personen gelegt wird. SST-Merkmale beinhalten Informationen zu Sprachrhythmus, Akzent und Betonung. Hierbei sind menschliche Stimmen hochgradig individuell und können von anderen Menschen nach nur wenigen Sätzen einem Sprecher zugeordnet werden, egal ob dieser zuvor bekannt oder unbekannt war. Nachforschungen haben sich lange Zeit auf FBA-Merkmale bezogen [2].

4.3 Transformer

Transformer kommen aus dem Bereich der maschinellen Übersetzung von Sprachen und wurden später für Natural Language Processing (NLP), bei welchem Daten sequenziell bearbeitet werden, adaptiert. Hierbei benutzt diese Form von Neuralem Netzwerk einen Self-Attention-Mechanismus, um jedem Element einer Sequenz Gewicht und Relationalität zuzuschreiben [16]. Wir empfehlen weiterführend die Arbeiten von Osswald et al. [17] und das Buch von Prince [18] für ein noch tieferes Verständnis.

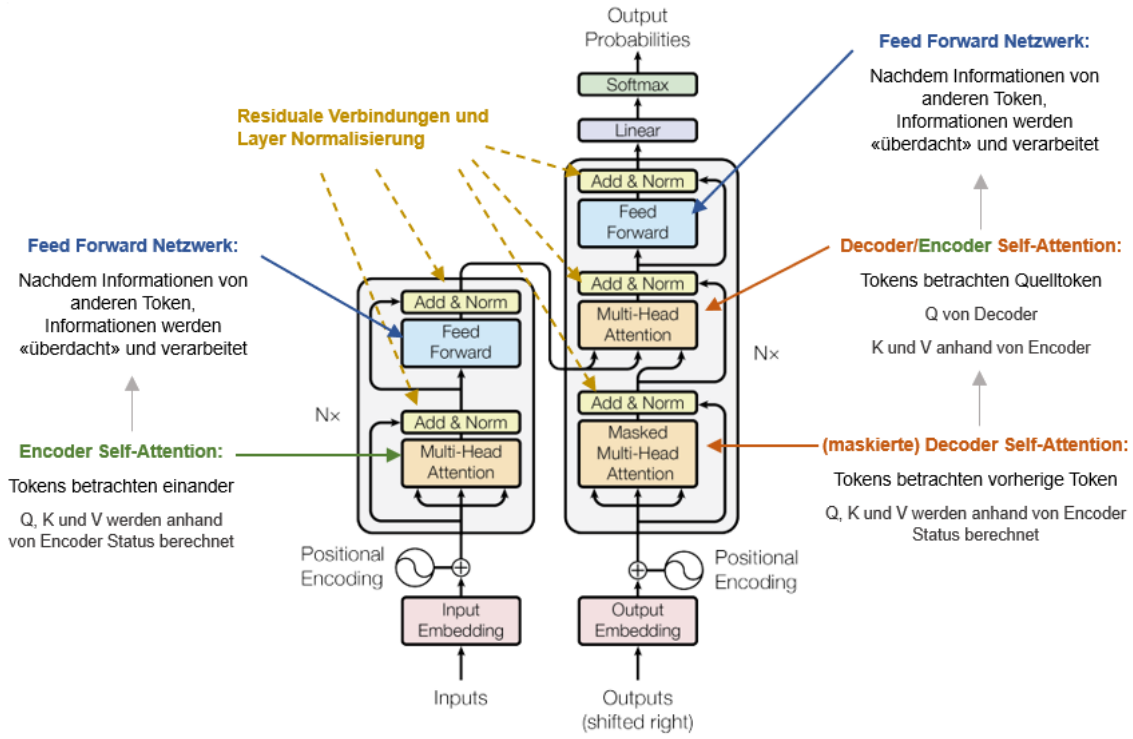


Abb 7 Visualisierung des Transformer-Modell abgeändert von [19]

4.3.1 Self Attention

Die Attention ist formal gegeben als:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

Wobei Q, K, V in der Regel lineare Projektionen aus den Eingabewerten mit je einer eigenen Gewichtsmatrix sind. Das d unter dem Bruchstrich ist die Embedding-Dimension, oft auch versteckte Dimension genannt und ist ein skalarer Wert. Damit wird verhindert, dass es zu nummerischen Problemen mit zu grossen Zahlen kommt.

Die Attention Logits berechnen sich aus QK^T. Diese werden durch einen Softmax Layer geführt (vgl. Kapitel 4.3.4) und anschliessend noch mit V berechnet. Die Gewichtsmatrix von V kann sich in der Embedding-Dimension unterscheiden. Die Self-Attention hat eine quadratische Komplexität.

4.3.2 Positional Encoding

Beim vorgeschlagenen Transformer von Vaswani et al. [16] ist der Attention Mechanismus Permutation Invariant. Dies bedeutet, dass unabhängig der Reihenfolge der Eingabe das gleiche Ergebnis erzielt wird. Aus diesem Grund kommt ein Positional Encoding zum Einsatz. Der Transformer lernt für den gleichen Token verschiedene Positionen, welche unterschiedlich behandelt werden können. Hierdurch ist der Transformer in der Lage positionelle Relationen zwischen den einzelnen Token in Betracht zu ziehen. Positional Encoding ist formal gegeben als:

$$PE(pos,\ 2i) = \sin\left(\frac{pos}{10000^{2i\ /\ d_{model}}}\right)$$

$$PE(pos,\ 2i+1) = \cos\left(\frac{pos}{10000^{2i\ /\ d_{model}}}\right)$$

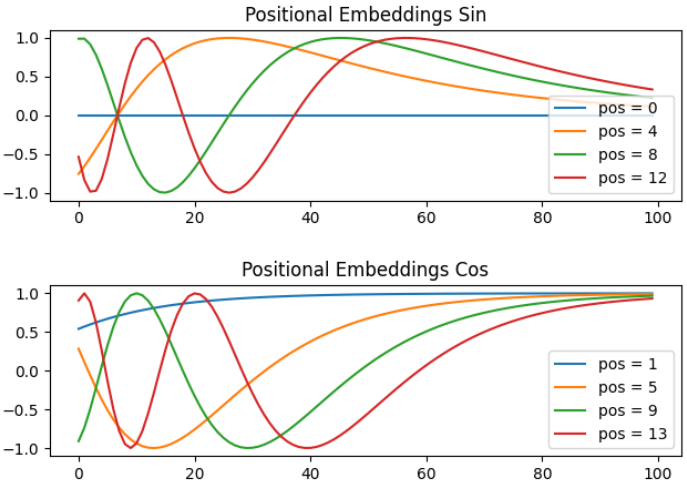


Abb 8 Visualisierung der Funktionen, welche für Positional Encoding genutzt werden

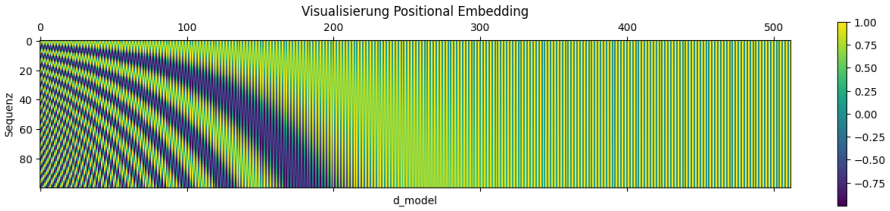


Abb 9 Visualisierung von Positional Encoding mit d_{model}=512 für 100 Werte

4.3.3 Multi-Head Attention (MHA)

Es wird spekuliert, dass die MHA den Attention Mechanismus robuster macht [18]. Dabei wird die Embedding-Dimension von Q, K, V durch H Köpfe unterteilt. Für jeden Kopf wird die Attention berechnet, welche am Ende wieder vertikal verknüpft werden. Jede Attention aus jedem Kopf kann als unabhängige logische Matrix betrachtet werden [16]

$$MultiHead(Q,\ K,\ V) = Concat(head1,\ ...,\ headh)W^O$$

4.3.4 Softmax

Softmax beschreibt eine Funktion aus der Stochastik, welche im Rahmen von DNN als Aktivierungsfunktion verwendet wird.

$$softmax(x_i) = \exp(x_i) \,/\, \sum_{j=1}^{N} \exp(x_j)$$

Die Rückgabewerte liegen auf einem Intervall von 0 bis 1. Es handelt sich dabei um eine nicht lineare Wahrscheinlichkeitsverteilung. Sie kann sowohl beim Forward-Pass als auch bei der Brackpropagation effizient ausgerechnet werden. Die Summe aller $softmax(x_i)$ ist 1. Im Transformer führt dies dazu, dass sich die Attention Logits entlang einer Dimension zu 1 aufsummieren lassen. Andere Aktivierungsfunktionen sind theoretisch denkbar.

4.3.5 Addition & Layer Normalization (ALN)

Die Resultate jedes vorhergehenden Modules (zum Beispiel MHA) werden elementweise zur Eingabe des Modules hinzugefügt. Hierdurch wird eine residuale Verbindung zwischen Ein- und Ausgabe erzielt. Die Eingabe kann hiermit direkten Einfluss auf die Ausgabe nehmen [16, 20, 21].

$$\mu = mean(x, axis = -1)$$
$$\sigma = std(x, axis = -1$$
$$Layernorm(x) = \gamma * (x - \mu) \,/\, \sigma + \beta$$

«Axis = -1» referenziert die versteckte Dimension des Eingabe Tensors. γ und β sind lernbare Parameter. Diese erlauben eine Skalierung respektive Verschiebung.

4.3.6 Position-wise Feed Forward (FFN)

Feed Forward-Netzwerke (FFN) dienen dazu die Kapazität von Modellen in Hinsicht des Lernens von komplexen Mustern und Relationen zu erhöhen. Um dies zu erreichen, werden nicht-lineare Transformationen an jeder Position der Sequenz genutzt [16].

$$FFN(x) = Activcation(x * W1 + b1) * W2 + b2$$

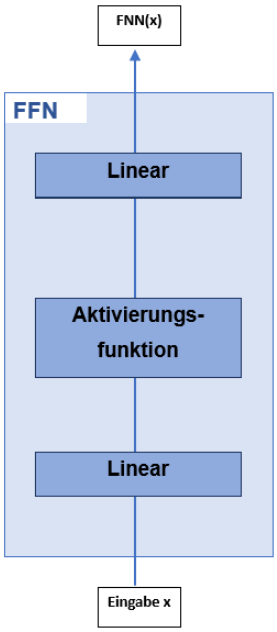


Abb 10 Visualisierung FFN Layer

4.3.7 Encoder/Decoder

Encoder können beliebig aneinandergereiht werden, bevor der Output an den Decoder übergeben wird. Ähnlich kann auch der Decoder beliebig aneinandergereiht werden. Ein Block besteht aus zwei Modulen, MHA und FF. Das FF hat nur einen Hidden Layer. Nach jedem Modul findet eine ALN statt. Die Module sind über residuale Verbindungen miteinander verknüpft. Im Gegensatz zum Encoder hat der Decoder zwei MHA-Module. Beim ersten MHA-Modul wird, anders als bei der klassischen Attention, die Eingabe maskiert. Diese Maskierung versteckt Informationen in der Attention. Dabei wird oftmals einfach die obere Diagonale der Attention Logits maskiert. Das zweite MHA-Modul wird Encoder/Decoder oder auch Shared Attention Modul genannt. Bei diesem Modul wird das Q aus dem Decoder übernommen, während K und V dem letzten Encoder-Block entstammen (vgl. Abb 7). [16]

4.3.8 Mean Squared Error Loss

Mean Squared Error Loss (MSE Loss) beschreibt eine Loss-Funktion, welche bei autoregressiven Ziel zum Einsatz kommt. Bei dieser wird der durchschnittliche Unterschied im Quadrat der vorhergesagten und realen Werten berechnet. Sie ist hierbei formal gegeben als:

$$mse = \frac{1}{n} * \sum (y - \hat{y})$$

4.3.9 Additive Margin Softmax

In Umlauf gebracht durch Wang et al. [22] beschreibt AM-Softmax eine Loss-Funktion, welche ursprünglich für Gesichtserkennung entwickelt wurde. Anders als die reguläre Softmax Funktion, eignet sich AM-Softmax dafür, Merkmale gleicher Klassen kompakter zu gestalten. Hierbei wird die Grenze zwischen zwei Klassen ausgedehnt und Merkmale einer Klasse in Richtung des Klassenzentrums zusammengestaucht. Dies erhöht die Diskrimination zwischen Merkmalen.

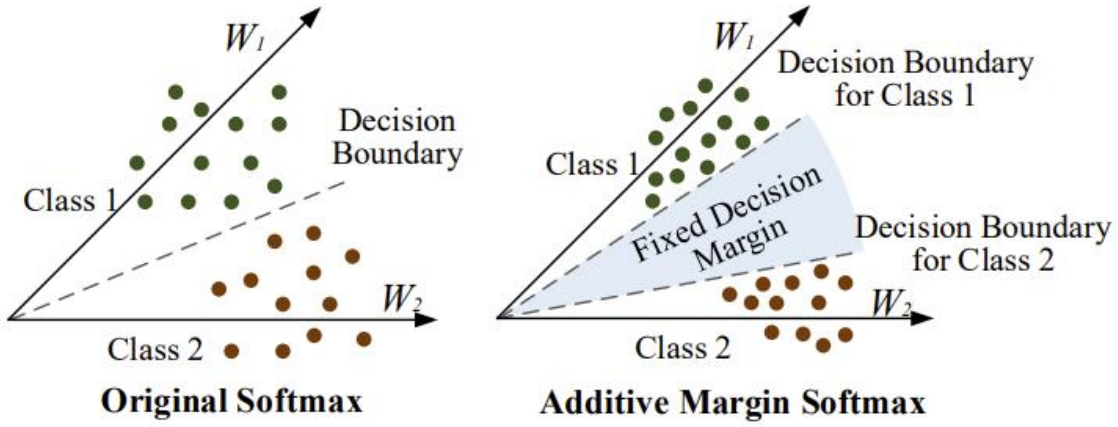


Abb 11 Vergleich Klassifizierung von Elementen durch Softmax und AM-Softmax entnommen aus [22]

4.3.10 Cosinus-Gleichheit

Die Cosinus-Gleichheit misst die Ähnlichkeit von zwei Vektoren. Die Cosinus-Gleichheit ist hierbei definiert als

$$cosineSimilarity(A, B) := \frac{A \bullet B}{\|A\| * \|B\|} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Wobei A und B n-dimensionale Vektoren sind und A_i, respektive B_i Komponenten dieser jeweiligen Vektoren. Der Wertebereich liegt zwischen [-1,1] wobei 1 eine absolute Gleichheit suggeriert und -1 absolute Ungleichheit.

4.3.11 ERR

Die Equal Error Rate ist definiert als der Wert, an welchem die Akzeptanzrate von falschen Proben (FAR) und die Ablehnungsraten wahrer Proben (FRR) gleich ist. Um diesen Wert zu erreichen, wird ein optimaler Schwellenwert gesucht, bei dem diese Balance erfüllt wird.

4.4 Verwandte Forschung

Transformer erzielen vielversprechende Resultate im Bereich der SV. So wurden diverse Transformer analysiert und implementiert. Sang et al. [23] und Zhang et al. [24] benutzen eine Kombination aus Convolution und Transformer. S-Vector [25] verfolgt einen anderen Ansatz und nutzt eine Architektur ähnlich zu X-Vector [26], einfach mit einem Transformer anstelle des TDNN. Alle verwenden zur Gewinnung der Sprecher Embeddings unterschiedliche Pooling Methoden.

Dosovitskiy et al. [27] schlägt vor, Bilder in Form von Patches zu trainieren. Darauf basierend beschreiben Gong et al. [28], Blade et al. [6] und Chong et al. [5] einen Algorithmus, um self-supervised mit Spektrogrammen Pretraining zu betreiben. Gong et al. [28] untersucht dabei den Unterschied zwischen Patches und Frames und erkennt, dass Frames zum besseren Ergebnis für die SV führen. Wang et al. [29], schlägt unterschiedliche Methoden zum Gewinnen der Sprecher Embeddings mit einem Transformer vor.

Unabhängig davon entwickelt Neururer et al. [3] eine Methode, wie man die SST-Ausnutzung misst.

Wir wenden Gongs et al. [28] Trainingsmethodik und Neururers [10] Test zum Überprüfen der SST-Ausnutzung an.

5 Methodik

5.1 Entfernen von SST-Merkmalen

Die Methodik von Neururer [10] wurde verwendet, um Modelle zu trainieren und zu evaluieren. Um ein Originalsegment (OS) zu erhalten, wird ein Segment aus der Originalaussage ausgeschnitten. Durch das Durchmischen des OS wird ein Mischsegment (SS) generiert. Die trainierten Modelle werden mittels der ERR miteinander verglichen. [3, 10]

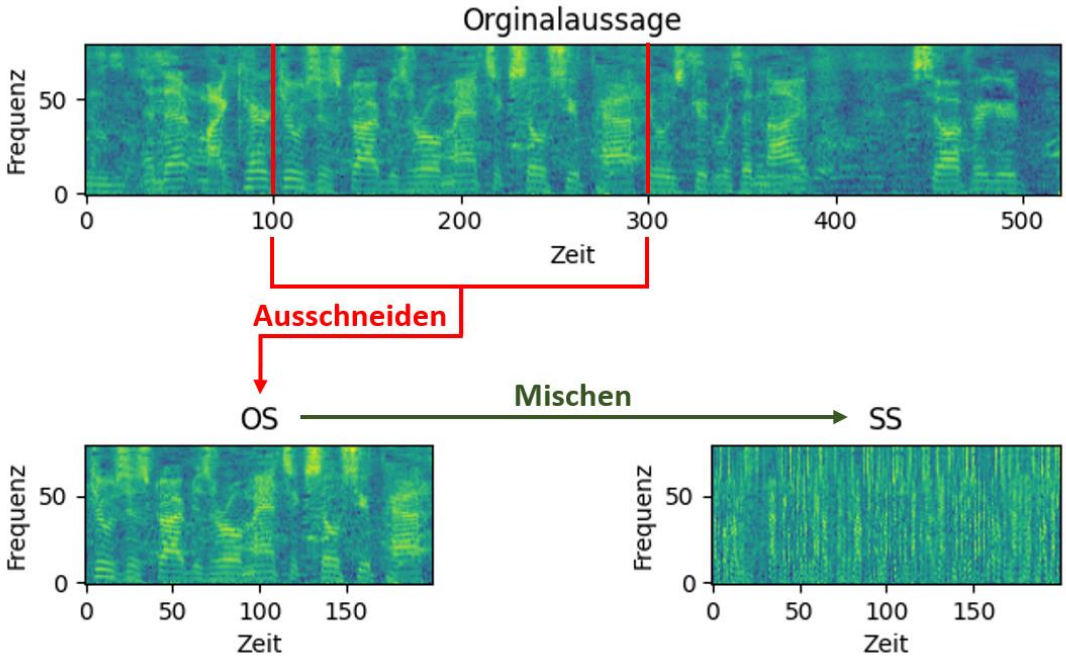


Abb 12 Ausschnitte aus Spektrogrammen zur Erzeugung von OS und SS

5.2 Vision Transformer

Transformer haben sich bei der Modellierung von langen kontextabhängigen Sequenzen bewährt [27]. Einige Transformer sind besser dazu geeignet Spektrogramme zu verarbeiten als andere. Der VIT hat die Möglichkeit globale Zusammenhänge zu verstehen und wird für die Bildverarbeitung genutzt [27]. Dieses Konzept eignet sich für Mel-Spektrogramme, da sie im entferntesten auch Bilder sind [28].

Für die Auswertung der SST-Ausnutzung wurde ein VIT-Modell basierend auf der Forschung von Baade et al. [6] ausgewählt. Hierbei wurde ein CLS-Token, ähnlich wie bei BERT [30] und Wang et al. [29] verwendet.

6 Experimente

6.1 Datenverarbeitung

Die Daten des VoxCeleb1&2 Datensatzes [31, 32, 8] wurden verwendet und mittels SpeechBrain [33] in einem CSV aufgelistet.

Parameter zur Erzeugung von Mel-Spektrogramm	
Koeffizient Pre-Emphasis (α)	0.97
Sample Rate	16'000
Segmentlänge Ausschnitt	2s
Grösse FFT (n_fft)	512
Fensterbreite (win_lenght, M)	25ms
Hop-Länge (hop, h)	9.6ms
Anzahl Filterbänke (n_mels)	80

Tabelle 1 Parameter Mel-Spektrogrammerzeugung

Die Audiodaten wurden durch eine zeitdiskrete STFT mit Hamming-Fensterfunktion unter Einsatz von Mel-Filterbänken in Mel-Spektrogramme konvertiert. Eine Aufteilung in 80 Filterbänke erfolgte. Des Weiteren wurde eine Pre-Emphasis mit α von 0.97 angewendet. 512 wurde als Anzahl der Samples (n_fft) der STFT gewählt.

Die gewonnenen Mel-Spektrogramme wurden nach Neururer [10] in SS und OS aufgeteilt.

6.2 Vision Transformer (VIT)

Parameter VIT	
Bildgrösse (img_size)	(80, 208)
Patch-Grösse (patch_size)	(80, 1)
Eingabekanäle (in_chans)	1
Embedding-Dimension (embed_dim)	192/384
Anzahl Encoder-Blöcke (depth)	12
Anzahl Köpfe Encoder (num_heads)	3
Embedding-Dimension Decoder (decoder_embed_dim)	512
Anzahl Decoder-Blöcke (decoder_depth)	8
Anzahl Köpfe Decoder	16
Rate MLP (mlp_ratio)	4/3
EPS	$1e^{-6}$
Optimizer	AdamW
Batch Size	2000
Scheduler	Cosinus
Lernrate	0.001
Weight Decay	0.05
Warm-Up Steps	5/10/20
Loss-Function	mse/am-softmax

Tabelle 2 Parameter genutzt in VIT (durch / getrennte Werte werden in verschieden Experimenten benutzt)

Die Grösse der Bildeingabe ist durch die Dimensionen der genutzten Mel-Spektrogramme gegeben. Basierend auf deren Dauer und Anzahl der verwendeten Filterbänke folgt die Verwendung von (80, 208). Als Embedding-Dimension dient 192 für das small und 384 für das medium Modell. Die Dimension des FFN ist das Produkt aus der MLP-Rate und der Embedding-Dimension. Daraus folgt, dass die FFN Dimension 768 für das Modell small und 1152 für das Modell medium ist. Es werden jeweils die Gewichte der 100 besten Train Losses in einer Epoche als Checkpoint gespeichert.

6.3 Maskierung

Die erzeugten Spektrogramme werden maskiert. Diese Maskierung wird sowohl beim Pretraining wie auch beim Finetuning angewendet. Hierbei werden 0 – 20 Frames in der Zeit- & Frequenzdimension maskiert. Zusätzlich werden während des Pretrainings noch 75% der Frames maskiert [6].

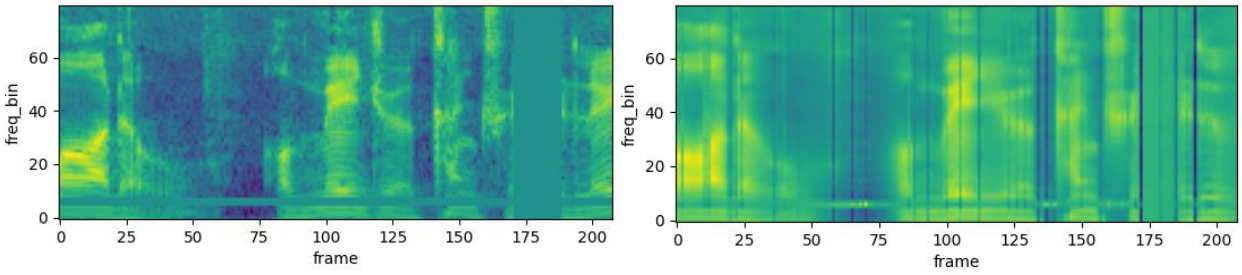


Abb 13 Vergleich Spektrogramm (rechts) und rekonstruiertes Spektrogramm (links)

6.4 Notation

Bei den Tabellen gilt die Zeilenbeschriftung als «trainiert auf», die Spaltenbeschriftung als «ausgewertet auf». In Grafiken und Texten wird «trainiert/ausgewertet» genutzt, um auszudrücken, auf welcher Variante die Daten trainiert respektive ausgewertet worden sind. Ein «P» angehängt sagt aus, dass ein autoregressives Pretraining verwendet wurde.

6.5 Training Setup

Es wurde eine Batch-Size von 50 verwendet, wobei 40 Batches akkumuliert wurden. Insgesamt wurden 80 Epochen lang für das Finetuning trainiert, wobei 5 Warm-Up Steps zum Einsatz kamen. Für das Pretraining wurde ein herkömmlicher MSE-Loss verwendet und für das Finetuning die AM-Softmax. Das Finetuning fand auf dem VoxCeleb1 Datensatz statt. Alle Resultate wurden auf dem VoxCeleb1 Verifikationssplit mittels Cosinus Gleichheit evaluiert. [34]

6.6 Experiment VIT 1(small)

Die Anzahl Parameter belaufen sich auf 5.6 Millionen beziehungsweise 30.9 Millionen beim Pretraining. Für das Pretraining wurden 20 Warm-Up Steps genutzt und über 40 Epochen lang trainiert. Ansonsten sind alle Parameter analog zur Variante ohne Pretraining.

6.6.1 Resultate

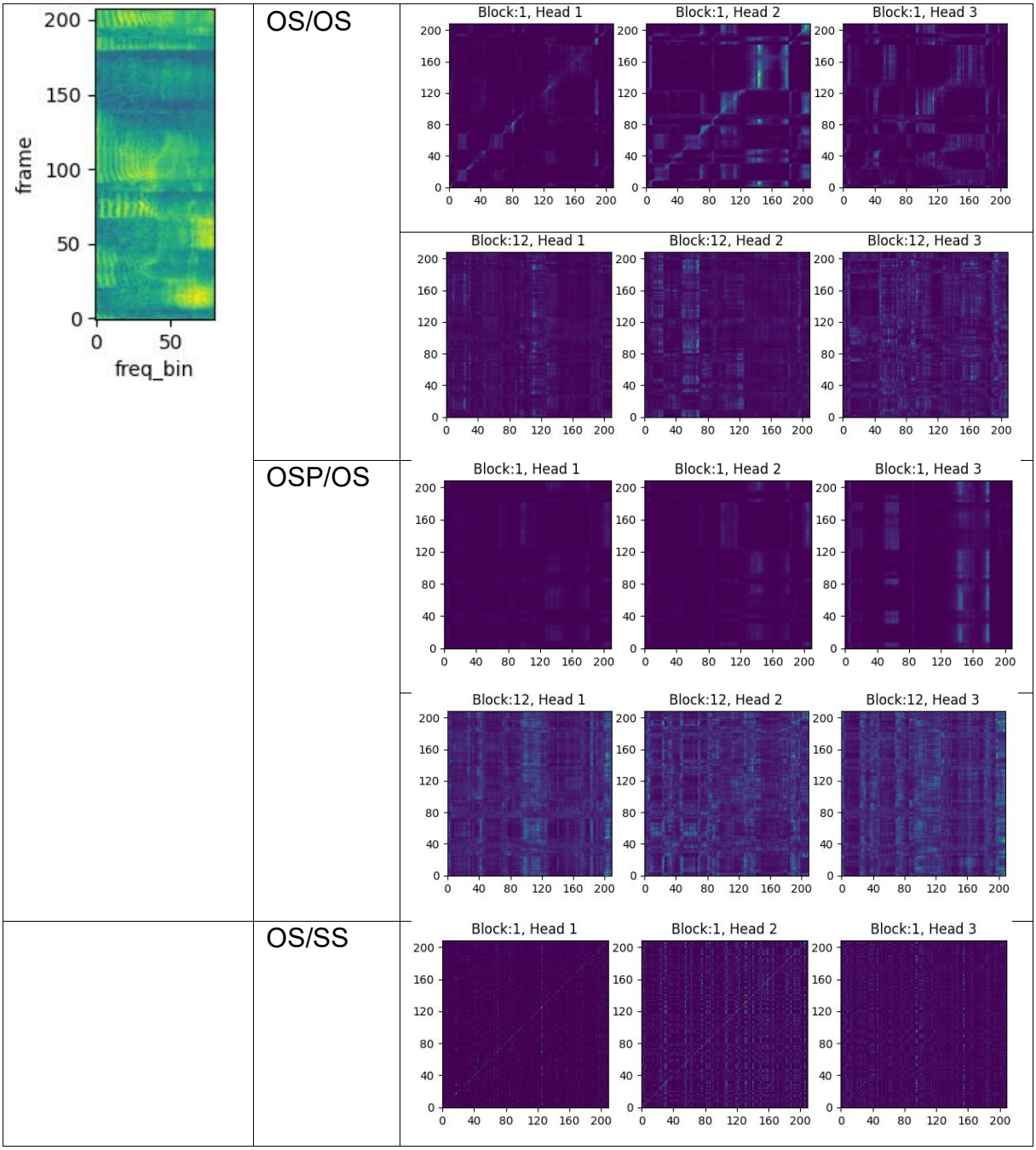
EER	OS	SS
OS	20.8	20.7
SS	20.4	20.5
OSP	17.8	20.0

Tabelle 3 ERR des Experiment VIT 1 - Oben: Testdaten, links: Datensatz, auf welchem Training stattfand

Ohne Pretraining (OS/OS, OS/SS, SS/OS, SS/SS) zeigen die Ergebnisse, dass das Modell sich lediglich auf die Eigenschaften der Frames stützt. SST-Merkmale werden ignoriert. Mit Pretraining (OSP/OS und OSP/SS) werden die SST-Merkmale etwas besser erkannt. Die EER ist niedriger beim Modell mit Pretraining (vgl. Tabelle 3).

6.6.2 Was sieht der Transformer:

Die Attention Logits in den früheren Blöcken konzentrieren sich eher auf lokale Merkmale, während sich die späteren Blöcke auf globale Merkmale fokusieren. Für OS/SS ist kein deutliches Muster erkennbar, während für OS/SS8 eine Konzentration auf globale Attention Logits in den späteren Schichten erkennbar ist.



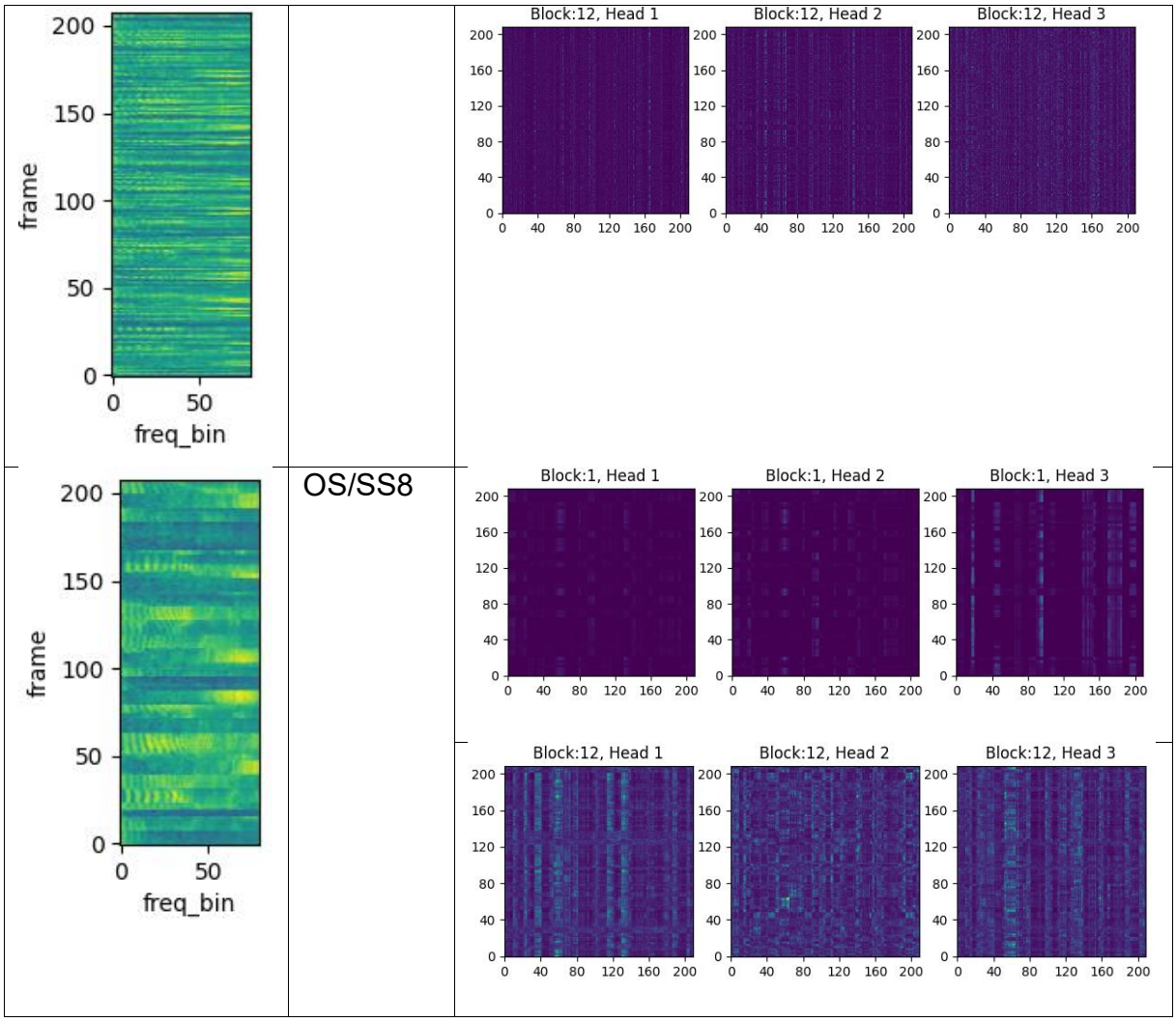


Tabelle 4 Vergleich Modelle und Attention

6.6.3 Lernverlauf

Anhand des Lernverlaufes (vgl. Abb. 14) ist zu erkennen, dass Pretraining dem Transformer hilft, schneller zu lernen.

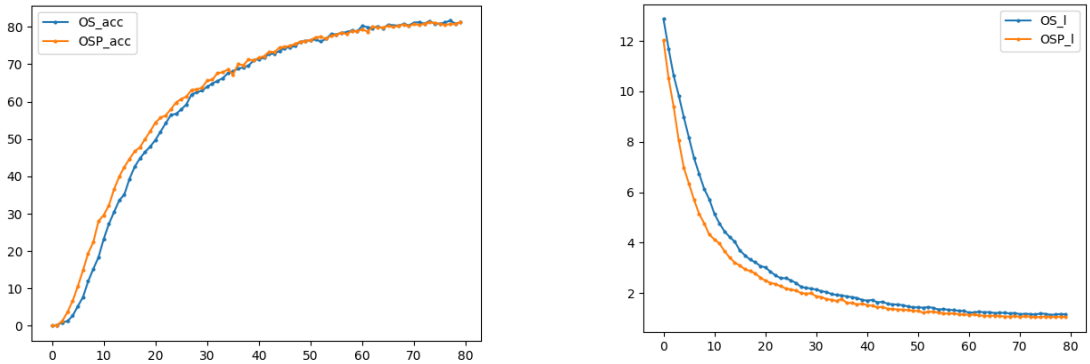


Abb 14 Lernverlauf Experiment 2 - ACC und l, wobei OSP auf Einsatz von Pretraining verweist

6.7 Experiment VIT 2 (medium)

Die Anzahl Parameter belaufen sich auf 21.9 Millionen respektiv auf 47 Millionen mit Pretraining. Beim Pretraining wurden für eine Dauer von 20 Epochen mit 10 Warm-Up-Steps trainiert. Hierbei wurde VoxCeleb1&2 als Datensatz verwendet. Mehr Pretraining wäre denkbar gewesen, wurde jedoch aufgrund von zeittechnischen Einschränkungen nicht durchgeführt. Für das Finetuning wurden insgesamt 120 Epochen trainiert. Die restlichen Parameter sind gleich des vorherigen Experimentes (vgl. Kapitel 6.6).

6.7.1 Resultate

EER	Epochen	OS	SS
OSP	60	16.6	19.4
OSP	120	18.2	19.7

Tabelle 5 Experiment 2 nach Finetuning - Vergleich ERR/Epochen, sowie SST-Variante

Wenn wir OSP/OS nach 60 Epochen mit OSP/OS nach 120 Epochen vergleichen, deutet dies darauf hin, dass der Transformer die SST-Merkmale verlernt. Insgesamt werden die SST-Merkmale nicht signifikant berücksichtigt. Dieses Modell schneidet nach 60 Epochen (OS/OS) besser ab als der VIT small. Umgekehrt erzielt VIT medium nach 120 Epochen (OS/OS) die schlechteren Ergebnisse als VIT small.

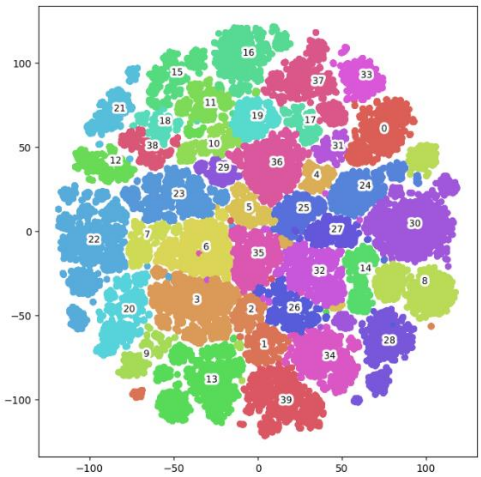


Abb 15 TSNE - Klassifizierung von 40 Sprechermodell medium mit Pretraining von 60 Epochen

7 Schlussfolgerung

7.1 Zusammenfassung

Neururers Test [3] konnte beim Pretraining nicht angewendet werden. Aus welchem Grund dies nicht gelungen ist, muss genauer untersucht werden. Es wurde versucht, das SS-Spektrogramm zu rekonstruieren.

An Experiment VIT 1 sieht man, dass das Modell ohne Pretraining nur FBA-Merkmale lernt. Dies kann an der Architektur oder am autoregressiven Ziel liegen. Es zeigt sich, dass auf SS trainierte Modelle [3] besser oder gleich gut abschneiden wie die auf OS trainierten Modelle. Dies deutet darauf hin, dass der Transformer falsch angewendet nur FBA-Merkmale lernt.

Bereits vielversprechender sind die vortrainierten Modelle von Experiment VIT 1 und VIT 2, wo ein kleiner Unterschied, zwischen den auf OS und SS evaluierten Modelle erkennbar ist. Der Test ist jedoch unvollständig. Der SS-Test müsste noch nachgeliefert werden, dann könnte man bestätigen, dass der Transformer nicht nur FBA-Merkmale lernt, sondern wirklich SST-Merkmale versteht.

An Experiment VIT 2 sehen wir, dass der Transformer schlecht generalisiert. Es wird zusätzlich vermutet, dass der Transformer dabei SST-Merkmale verlernt, da die OS und SS Metriken bei 120 Epochen näher aneinanderrücken. Deshalb denken wir, dass eine Encoder/Decoder Architektur oder eine andere Architektur beim Finetuning besser geeignet ist, um SST-Merkmale zu erkennen.

Insgesamt erkennt unser Modell die SST-Merkmale nicht signifikant anders als die bereits getestete Neuronale Netzwerke durch Neururer et al. [3].

7.2 Zukünftige Arbeiten

Neururers Test [3] muss so angewendet werden, dass er auch beim Pretraining verwendet werden kann. Zum Beispiel könnte man versuchen, immer das ursprüngliche Mel-Spektrogramm zu rekonstruieren.

Zudem sind die hier erzeugten Mel-Spektrogramme nicht optimal. Wir vermuten das eine kürzere Hop-Länge dazu führen könnte, dass SST-Merkmale besser zu erkennen sind.

Die Experimente sollten mehrfach und mit konsistenten Hyperparametern wiederholt werden, um definitive Aussagen treffen zu können.

Auch soll ein Finetuning auf die mit OS vortrainierten Modellen auch mit die SS-Spektrogramme angewendet werden und andersherum.

Der VIT soll weiter mit Neururer [3] untersucht werden. Vorgeschlagen sind Optimierungen wie LayerScale [35], Joint Losses [28], eine effizientere Gewinnung der Sprecher Embeddings [6, 29], sowie das Variieren der Anzahl der Köpfe in der MHA. Es sollen weiterhin unterschiedliche Transformer Architekturen untersucht werden, wie zum Bespiel das klassische Encoder/Decoder Modell aber auch andere Modelle [36].

8 Danksagung

Wir wollen uns vielmals bei Dr. Thilo Stadelmann für seine Betreuung und das Beantworten von Fragen bedanken. Des Weiteren danken wir Ramona Egli für das Korrekturlesen dieser Arbeit.

9 Anhang

9.1 GitHub README

Vision Transformer[1]

An implementation of Vision Transformer for masked Spec self-training in MAE Style

Requirements:

- Python 3.9

Installation

```
pip install -r requirements.txt
```

Usage:

Test trained Model

There are already pretrained Models to use, which can be found here google drive.

The command for testing is:

```
python3 train.py --modul_name=VisionTransformer --model_size=small --shuffle_type=os --test --checkpoint="Path to Model"
```

For evaluation we use the veri_test2.txt, which can be found under the "files" folder.

Train your model

```
python3 train.py --modul_name=VisionTransformer --model_size=small --shuffle_type=os --max_epochs=80
```

The above configuration is enough to rerun our experiments. There are many more parameters available. For these inspect the train.py in the root.

Prepare the data!

It is possible to use our voxceleb_prepare/voxceleb2_prepare in the "data" folder scripts to create the CSVs. Alternatively, we deliver the CSVs already under "files/linux" where you can simply replace a part of the path.

[1] Verfügbar unter https://github.com/SafuanA/VisionTransformer

Code

- The PytorchLightning Modul is directly implemented in the train.py
- The PyTorch Modul for our Experiments can be found under "Modul.VisionTransformer"
- The Spectogramm processing can be found under "utils.features"
- In the "data" folder you find our "DataModul.py" and the "dataset.py" used for our experiments
- The Code used for generating our Plots can be found under "plotting" or "utils.Analysis"
- In "utils" there are many more files: for example, calculating the cosine score, the amsoftmax loss, accuracy calculation, augmenting the data.

9.2 Aufgabenstellung

Typ der Arbeit

- Bachelorarbeit

Titel
Teaching Neural Nets to Learn Dynamic Voice Features for Automatic Speaker Verification

Beschreibung

Similar to many other use cases, deep learning has revolutionized automatic speaker verification in recent years. Superior results (some of them achieved and published by previous generations of ZHAW students during their PA) were often credited to the capabilities of such neural networks to capture temporal aspects of a voice like timbre or speaking melody. However, a recent study (again, by a ZHAW student) showed that deep learning systems for speaker verification all but ignore temporal features: they lazily model the easiest extractable patterns in the data, which happen to be mere frequency distributions. The scientific literature instead promises huge gains from being able to model frequency in its short-term temporal dynamics. In this BA, the goal is to for the first time build a machine learning system that explicitly models such supra-segmental features of a voice, by incorporating a suitable inductive bias into the learning system. One way to achieve this could be to use transformer models (made famous in the area of NLP and recently being used also for computer vision) and training them in a self-supervised way that enforces the exploitation of speaking dynamics (e.g., by predicting future segments from past samples). A scientific publication together with the supervisors is possible. Work packages: - Review of related work (scientific literature) - Define a suitable neural network architecture and learning task together with supervisors - Set up data and development environment (locally and on our GPU cluster) - Build initial prototype, iterate (focus on novelty and meaningful experimentation) - Write a scientific report with a focus on motivation, argumentation, methods, evaluation & results (the PA thesis)

Voraussetzungen

This thesis can be conducted in German or English. The students should have strong interest and potentially even basic knowledge in deep learning and constructing deep learning models in Python. Strong interest in AI and hands-on work to build something that lasts is required, as well as the ability to independent conceptual work, very good general programming skills and a pragmatic approach to development and experimentation. Top grades in the previous course of study are typically indicative of these skills. The Computer Vision, Perception and Cognition Group at the ZHAW Centre for Artificial Intelligence conducts research and teaching in pattern recognition using deep learning methodology. In our thesis proposals, we want to address high-achieving students with a curiosity for gaining (first) experience with scientific research applied to the problem at hand. The supervisors are very enthusiastic about the topics and have plenty of experience as well as detailed ideas (and offering of support) for the project startup. We are looking for equally enthusiastic students, with the hope of achieving results that can be published at a scientific conference.

Themen Betreuer

Stadelmann, Thilo

stdm@zhaw.ch

Hauptbetreuer

Stadelmann, Thilo

stdm@zhaw.ch

Primäres Fachgebiet

Artificial Intelligence

Weitere Fachgebiete

- Artificial Intelligence
- Bildverarbeitung
- Datenanalyse
- Digitale Signalverarbeitung
- Machine Learning
- Software

Studiengang
- Informatik
- Systemtechnik

Institut / Zentren
- Centre for Artificial Intelligence (CAI)

Interne Partner
- Centre for Artificial Intelligence (CAI)

Informationslink
- https://stdm.github.io/research/#voice

Bemerkungen

A recent publication (under review) outlines the problem of learning supra-segmental temporal features and can be made available upon request.

Deutsch: JA, English, JA

9.3 Abbildungsverzeichnis

9.4 Tabellenverzeichnis

9.5 Referenzen

[1] A. K. N. T. S. a. I. S. Radford, „Improving language understanding by generative pre-training,“ 2018.

[2] M.-J. K. V. D. Adrian Leemann, „Speaker-individuality in suprasegmental temporal features: Implications for forensic voice comparison,“ Forensic science international, 238, 2014, pp. 59-67.

[3] V. D. T. S. Daniel Neururer, Deep Cheating: Deep Neural Networks for Automatic Speaker Recognition Do Not (Yet) Learn Supra-Segmental Temporal Features, Submitted for Review, 2023.

[4] R. Wightman, „PyTorch Image Models,“ Github, 2019.

[5] H. W. P. Z. Q. Z. Dading Chong, „Masked spectrogram prediction for self-supervised audio pre-training.,“ In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2023, pp. 1-5.

[6] P. P. D. H. Alan Baade, MAE-AST: Masked Autoencoding Audio Spectrogram Transformer, arXiv preprint arXiv:2203.16691, 2022.

[7] L. a. L. J. a. H. J.-T. a. Y. K. a. Y. D. a. S. F. a. S. M. a. Z. G. a. H. X. a. W. J. a. G. Y. a. A. A. Deng, „Recent advances in deep learning for speech research at Microsoft,“ IEEE, in: Proc. ICASSP, 2013, p. 8604–8608.

[8] A. N. A. Z. Joon Son Chung, VoxCeleb2: Deep Speaker Recognition, INTERSPEECH : arXiv preprint arXiv:1806.05622 , 2018.

[9] P. P. Y. L. A. Y. N. H. Lee, „Unsupervised feature learning for audio classification using convolutional deep belief networks,“ in: Proc. NIPS, 2009, p. 1096–1104.

[10] D. Neururer, Exploiting the Full Information of Varying Length Utterances for DNN-Based Speaker, Winterthur: ZHAW, 2022.

[11] M. Huzaifah, Comparison of Time-Frequency Representations for Environmental Sound Classification using Convolutional Neural Networks, arXiv preprint arXiv:1706.07156, 2017.

[12] M. .. S. M. Ankita S. Chavan, Effect of Pre-processing along with MFCC, IJEDR, Volume 4, Issue 3, 2016.

[13] J. W. T. J. W. Cooley, „An algorithm for the machine calculation of complex Fourier series," Bd. 90, pp. 297-301, 1965.

[14] Phonical, „commons.wikimedia.org," , 30 11 2017. [Online]. Available: https://commons.wikimedia.org/wiki/File:FFT-Time-Frequency-View.png. [Zugriff am 07 06 2023].

[15] A. A. C. M. P. G. S.B. Dhonde, „Performance Evaluation of Mel and Bark Scale based Features for Text-Independent Speaker Identification," International Journal of Innovative Technology and Exploring Engineering 8.11, 2019, pp. 3734-3738.

[16] N. S. N. P. J. U. L. J. A. N. G. L. K. I. P. Ashish Vaswani, Attention Is All You Need, Advances in neural information processing systems, 30., 2017.

[17] J. N. E. R. E. S. J. M. A. Z. A. &. V. M. von Oswald, „Transformers learn in-context by gradient descent," *arXiv preprint arXiv:2212.07677,* 2022.

[18] S. J. Prince, Understanding Deep Learnin, https://udlbook.github.io/udlbook/: MIT Press, 2023.

[19] R. Hum-Sah, „stats.stackexchange.com," 21 2 2022. [Online]. Available: https://stats.stackexchange.com/questions/565196/why-are-residual-connections-needed-in-transformer-architectures. [Zugriff am 07 06 2023].

[20] X. Z. S. R. a. J. S. Kaiming He, „Deep residual learning for image recognition.," In Proceedings of the IEEE Conference on Computer Vision and Pattern, 2016, p. 770–778.

[21] J. R. K. a. G. E. H. Jimmy Lei Ba, „Layer normalization," arXiv preprint arXiv:1607.06450, 2016.

[22] F. C. J. L. W. &. L. H. Wang, „Additive margin softmax for face verification,“ 25. Jg., Nr. 7, IEEE Signal Processing Letters, 2018, pp. 926-930.

[23] Y. Z. G. L. J. H. H. J. W. Mufan Sang, „Improving Transformer-based Networks With Locality For Automatic Speaker Verification,“ IEEE, In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2023, pp. 1-5.

[24] Z. L. H. W. S. Z. P. H. Z. W. H.-y. L. H. M. Yang Zhang, MFA-Conformer: Multi-scale Feature Aggregation Conformer for Automatic Speaker Verification, INTERSPEECH 2022, März 2022.

[25] S. U. a. S. V. K. N. J. M. S. Mary, "S-Vectors and TESA: Speaker Embeddings and a Speaker Authenticator Based on Transformer Encoder", IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 30, pp. 404-413, 2022.

[26] D. G.-R. G. S. D. P. S. K. David Snyder, „X-VECTORS: ROBUST DNN EMBEDDINGS FOR SPEAKER RECOGNITION,“ Bd. IEEE, Nr. In: 2018 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp. 5329-5333, 2018.

[27] L. B. A. K. D. W. X. Z. T. U. M. D. M. M. G. H. S. G. J. U. N. H. Alexey Dosovitskiy, An image is worth 16x16 words: Transformers for image recognition at scale., arXiv preprint arXiv:2010.11929, 2020.

[28] C.-I. J. L. Y.-A. C. J. G. Yuan Gong, „SSAST: Self-supervised audio spectrogram transformer.,“ In Proceedings of the AAAI Conference on Artificial Intelligence, 2022, pp. 10699-10709.

[29] J. A. L. Z. S. L. Z. W. T. K. Q. L. Y. Z. Rui Wang, „Multi-view self-attention based transformer for speaker recognition.,“ IEEE., In ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2022, pp. 6732-6736.

[30] M.-W. C. K. L. K. T. Jacob Devlin, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, arXiv preprint arXiv:1810.04805, 2018.

[31] J. S. C. A. Z. A. Nagrani*, VoxCeleb: a large-scale speaker identification dataset, INTERSPEECH, 2017.

[32] J. S. C. W. X. A. Z. A. Nagrani*, Voxceleb: Large-scale speaker verification in the wild, Computer Science and Language, 2019.

[33] T. P. P. P. A. R. S. C. L. L. C. S. N. D. A. H. J. Z. J.-C. C. S.-L. Y. S.-W. F. C.-F. L. E. R. e. a. Mirco Ravanelli, „SpeechBrain: A General-Purpose Speech Toolkit," Nr. arXiv preprint arXiv:2106.04624, 2021.

[34] VoxCeleb, „robots.ox.ac.uk," [Online].

[35] M. C. A. S. G. S. H. J. Hugo Touvron, „Going deeper with Image Transformers," In Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 32-42.

[36] H.-R. P. M. B. B. F. G. B. S. Hamza Keurti, Homomorphism Autoencoder -- Learning Group Structured Representations from Observed Transitions, arXiv preprint arXiv:2207.12067, 2022.