Integration einer Sprachsteuerungsfunktion in Mobile Apps

Themenbereiche:	Sprachsteuerung, Audio, Machine Learning, Triggerwort-Erkennung, Mobile Apps
Studierender:	Ruben Nuñez
Dozent:	Dr. Florian Herzog
Experte:	Damien Piguet
Auftraggeber:	Stefan Reinhard
Keywords:	Sprachsteuerung, Mobile Apps, Machine Learning, Triggerwort-Erkennung, Audioverarbeitung, Datensatzerstellung, Modelltraining, App-Integration

1 Aufgabenstellung

Das Ziel dieser Bachelorarbeit ist die Entwicklung einer Sprachsteuerungsfunktion für mobile Anwendungen, mit einem besonderen Fokus auf der Erkennung von Triggerwörtern aus akustischer Sprache. Die Arbeit umfasst die Entwicklung eines Machine Learning-Modells, dessen Integration in eine mobile Plattform, insbesondere iOS, sowie die Berücksichtigung von Datenschutz und ethischen Richtlinien bei der Erstellung und Nutzung des für das Training des Modells erforderlichen Datensatzes.

2 Ergebnisse

Im Rahmen der Bachelorarbeit wurden folgende Ergebnisse erzielt:

- Ein **Machine Learning-Modell**, das in der Lage ist, Triggerwörter in akustischer Sprache zu erkennen.
- Eine **Integration in eine iOS-App**, die das Modell verwendet, um Triggerwörter in Echtzeit zu erkennen.
- Ein **Datensatz**, welcher ethische Richtlinien und Datenschutzbestimmungen einhält, wurde erstellt und verwendet, um das Modell zu trainieren.
- Eine **Dokumentation**, die die Entwicklung des Modells, die Erstellung des Datensatzes und die Implementierung der App beschreibt.

Das Machine Learning Modell, benannt **WakeupTriggerConvLSTM2s**, ist eine Kombination aus einem *Convolutional Neural Network* (CNN) und einem *Long Short-Term Memory* (LSTM) Netzwerk. Das CNN extrahiert die akustischen Merkmale der Sprache, während das LSTM die zeitlichen Abhängigkeiten zwischen diesen Merkmalen erfasst. Es wurde in PyTorch implementiert, mit dem erstellten Datensatz trainiert und erreichte eine Genauigkeit (Accuracy) von 0.88 sowie einen F1-Score von 0.80 auf dem Testdatensatz. Zusätzlich wurde eine Transformationskomponente **AudioToSpectrogramTransformJit** in PyTorch entwickelt, die Audio in Spektrogramme umwandelt.

Die Effizienz des Modells, insbesondere im Hinblick auf Echtzeitfähigkeit, war entscheidend. Das Modell erreicht eine Vorhersagegeschwindigkeit von 0.01 Sekunden pro Spektrogramm, was den Anforderungen einer Echtzeiterkennung entspricht, ohne die Leistung des Geräts zu beeinträchtigen. Die entwickelten PyTorch-Module **AudioToSpectrogramTransformJit** und **WakeupTriggerConvLSTM2s** wurden in einer iOS-App implementiert, um eine effiziente Triggerworterkennung in Echtzeit zu ermöglichen.

Es wurde eine Recorder-Webseite für die Aufnahme von Sprachsamples entwickelt und auf der Google Cloud Platform gehostet. Diese Webseite ist eine dockerisierte Python Flask App die es ermöglichte eine effiziente Sammlung von Sprachdaten zu gewährleisten.

Zur Erhöhung der Diversität und Robustheit des Datensatzes kamen verschiedene Augmentierungstechniken zum Einsatz. Etwa 700 Sprachaufnahmen wurden von mehr als 10 Personen aufgenommen und durch Augmentierung auf 10.000 Samples erhöht. Der Datensatz beinhaltete neben dem Triggerwort 'Hey FOOBY' auch andere Wörter sowie Silence und Noise Samples, um das Modell unter verschiedenen Bedingungen zu trainieren und seine Reaktion auf Hintergrundgeräusche zu verbessern.

3 Lösungskonzept

In dieser Bachelorarbeit wurde innerhalb von 14 Wochen ein System zur Erkennung von Triggerwörtern in akustischer Sprache entwickelt. Ein zentraler Bestandteil des Systems ist die Analyse von Audio-Chunks, um Triggerwörter zu identifizieren. Dafür wird in Echtzeit der Buffer des Mikrofons ausgelesen und in Audio-Chunks verarbeitet. Der Buffer ist so konfiguriert, dass er in einem Intervall von 0.1 Sekunden ausgelesen wird. Die Grösse der Audio-Chunks beträgt dabei 2 Sekunden, das entspricht 32000 Samples bei einer Samplerate von 16kHz. Die nachstehende Abbildung visualisiert die Struktur dieser Lösung.

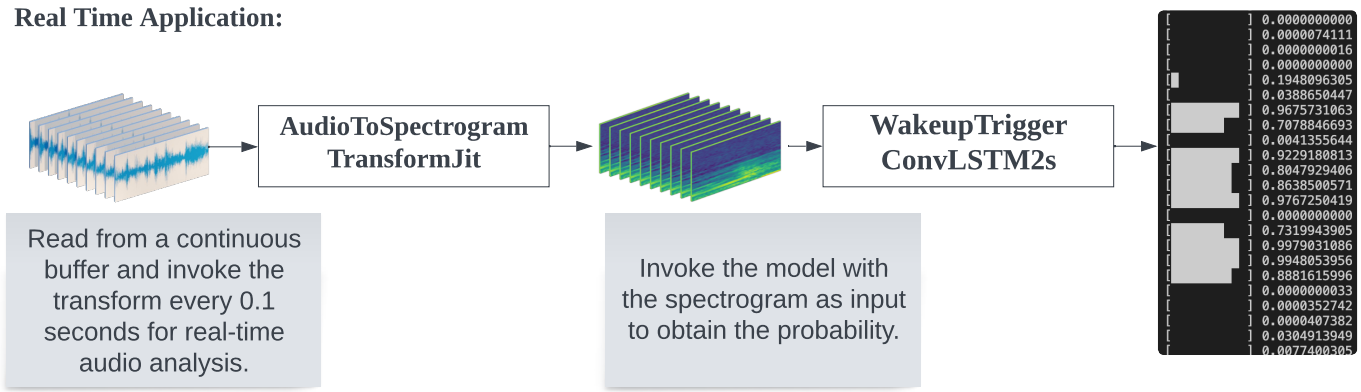

Abbildung 1: Schematische Darstellung der Real Time Anwendung

Die Erkennung basiert auf der Berechnung des gleitenden Durchschnitts (SMA) der Wahrscheinlichkeiten der letzten n Audio-Chunks. Überschreitet der Durchschnitt einen festgelegten Schwellenwert, wird das Triggerwort als erkannt betrachtet. Diese Methode sichert eine effiziente und präzise Erkennung. Die folgende Abbildung illustriert die Berechnung des SMA für verschiedene Anzahlen von Audio-Chunks.

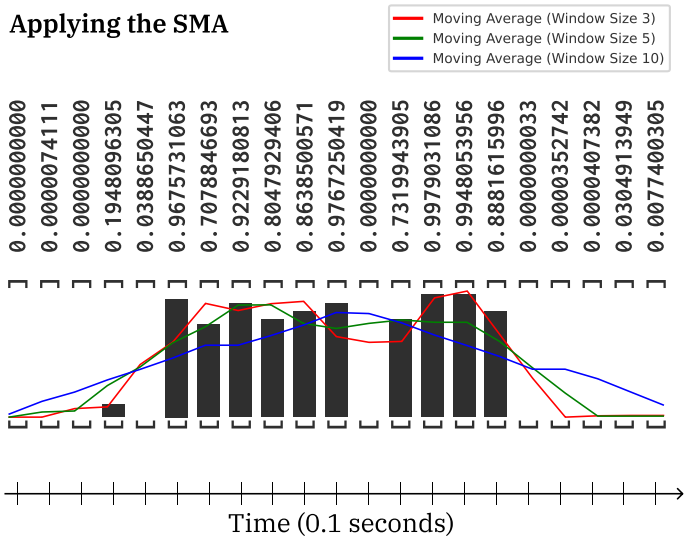

Abbildung 2: Darstellung der SMA-Berechnung über Audio-Chunks

Durch die Kalibrierung des Schwellenwerts sowie der Anzahl Audio-Chunks, die für die Berechnung des SMA verwendet werden, kann die Sensitivität der Erkennung angepasst werden.

Ebenfalls wurden User Tests durchgeführt, um die Benutzerfreundlichkeit der App zu testen. Sechs Personen nahmen an den Tests teil. Bei den Tests wurde in einer Konfusionsmatrix die Anzahl der korrekten und falschen Erkennungen der Triggerwörter aufgezeichnet. Die Tests zeigten, dass die Erkennung des Triggerworts in 50 von 61 Fällen erkannt wurde (True Positive). In 11 von 61 Fällen wurde das Triggerwort nicht erkannt (False Negative). Bei der other Klasse wurden

fälschlicherweise 28 von 63 Samples als Triggerwort erkannt (False Positive) und 35 von 63 Samples korrekt als other Klasse erkannt (True Negative). Die Konfusionsmatrix ist in der folgenden Abbildung dargestellt.

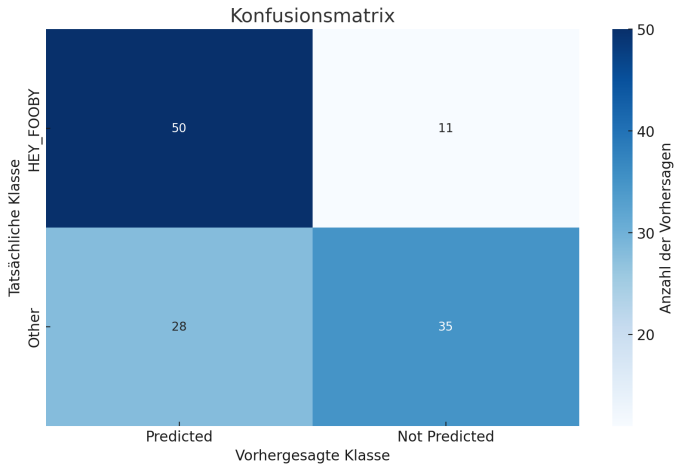

Abbildung 3: Konfusionsmatrix der User Tests

4 Spezielle Herausforderungen

Eines der Herausforderungen war es die erarbeiteten PyTorch Module über TorchScript zu konvertieren so dass diese in der iOS App verwendet werden können. Die Grundidee hinter TorchScript ist es, die PyTorch Module in ein Format zu konvertieren, das von der C++ API von PyTorch verwendet werden kann.

> TorchScript is a way to create serializable and optimizable models from PyTorch code. Any TorchScript program can be saved from a Python process and loaded in a process where there is no Python dependency. („PyTorch Just-In-Time Compiler (JIT) Documentation“, 2023)

Neben dem konvertieren musste auch die Integration in Objective-C, C++ und Swift erfolgen. Das Ganze ist dabei nicht so einfach wie es klingt. Es mussten einige Hürden überwunden werden, um die Module erfolgreich in die iOS App zu integrieren. Das Grundsätzliche Problem war, die fehlende Erfahrung in der Entwicklung mit Objective-C und C++.

5 Ausblick