Universiteit Utrecht

Emotieherkenning door Spraakherkenningssoftware

Rens Kersbergen

5853133

Taalwetenschap

Abstract

Het belang van het ontwikkelen van automatische spraakherkenning (ASR) wordt steeds groter. Vooruitgangen in neurale netwerken bieden de mogelijkheid om geavanceerde state-of-the-art spraakherkenningstechnieken toe te passen op spraakemotieherkenning. Emotie komt in het spraaksignaal voor in de vorm van stemkwaliteit, toonhoogte, formantfrequenties en spraaktempo. State-of-the-art Speech Emotion Recognition (SER) in Kaldi wordt geanalyseerd en vergeleken met een nieuw SER-experiment in Python om te ontdekken wat positieve en negatieve effecten heeft op de prestatie van het neuraal netwerk model. De conclusie is dat niet alle emotiecategorieën geschikt zijn als trainings- en testdata, dat perturbatie geschikt is voor data augmentatie, en dat een Time Delay Neural Network (TDNN) LSTM de meest geschikte architectuurontwerp is voor een SER systeem.

Introductie

Automatische emotieherkenning is het voorspellen van een emotie op basis van kenmerken die emotiecategorieën van elkaar onderscheiden. Deze kenmerken kunnen op verschillende manieren worden gemeten. In dit onderzoek worden emotiekenmerken gemeten in het spraaksignaal. Deze vorm van emotieherkenning staat beter bekend als Speech Emotion Recognition (SER) en is onderdeel van het onderwerp spraakherkenning. De vaardigheid om emotie te herkennen, kan geleerd worden door een kunstmatig neuraal netwerk gemaakt op moderne computers. Waar spraakproductie kan worden verricht met relatief eenvoudige computers, is spraakherkenning afhankelijk van de snelheid waarin berekeningen gedaan kunnen worden (O'Shaughnessy, 2008). Dat wil overigens niet zeggen dat spraakherkenning niet goed kan worden verricht met eenvoudige analoge computers. In 1952 bereikte het Audrey System een nauwkeurigheid van 97% zolang de opnames aan bepaalde specifieke eisen voldeden zoals een perfecte opnamekwaliteit, een select aantal klanken en één bepaalde stem (Pieraccini, 2012). Sindsdien gaan de vooruitgangen van spraakherkenning parallel met de vooruitgangen in computers en is het mogelijk om voor diverse opnamekwaliteiten, talen en stemmen emotie te herkennen.

SER kan worden toegepast op online educatie (Tickle, 2013), callcenters (Lee, 2008; Petrushin, 1999, Morrisson, 2007), videogames (Szwoch, 2015), gezondheidszorg (Torous, 2014; Hossain, 2015) en verkeersveiligheid (Kamaruddin, 2011; Grimm, 2007). Neem het voorbeeld van verkeersveiligheid. Dat SER indirect gerelateerd is aan het voorkomen van verkeersongelukken wordt duidelijk in de volgende redenering. SER maakt het verkeer veiliger omdat: emotie in het spraaksignaal zit (Davletcharova, 2015), de bestuurder wordt beïnvloed door emotie (Clore, 2007), en aangezien het gedrag van bestuurders een factor is bij ongelukken in het verkeer (Nass, 2005). Als we een verkeersongeluk kunnen voorspellen, dan kunnen we ook acties ondernemen om deze te voorkomen door bijvoorbeeld kalmerende

muziek aan te zetten. Hieruit kunnen we concluderen dat ontwikkelingen van SER bijdragen aan het voorspellen en voorkomen van ongelukken.

De huidige onderzoeksvraag “Hoe kan een spraakherkenningssoftware verbeterd worden in het herkennen van emotie?” wordt beantwoord in de vorm van een handleiding voor bestaande state-of-the-art (leidend in de ontwikkeling van een onderzoeksgebied) spraakherkenningssoftware Kaldi. Het doel van deze handleiding is het beschrijven van nieuwe oplossingen waarbij linguïstische kennis wordt gecombineerd met kennis over Deep Neural Networks (DNN) en wordt toegepast op SER. Hiermee worden problemen met SER opgelost en de prestatie van Neurale Netwerken op het voorspellen van emoties verbeterd.

Het onderzoek begint met een overzicht van voorgaande literatuur over emotiecategorieën en emotie in het spraaksignaal waarmee wordt gekeken naar hoe emotie wordt gerepresenteerd in het spraaksignaal. Daarna wordt er met ondersteuning van een uitleg over de werking van neurale netwerken en hun rol in SER uitgelegd hoe er tot nu toe is omgegaan met SER. Een eerder gedaan onderzoek over SER in Kaldi wordt als baseline genomen en vergeleken met een nieuw neuraal netwerk model. Een baseline model is een nulpunt waarmee de resultaten van een experiment worden vergeleken. De meest optimale emotiecategorieën, datasets, algoritmes en architecturen voor SER worden in dit Python experiment toegepast. Het onderzoek sluit af met een conclusie over wat werkt en wat beter kan in de huidige state-of-the-art SER.

In dit onderzoek worden veel afkortingen gebruikt die op verspreid over de tekst worden uitgelegd. Voor de overzichtelijkheid zijn alle afkortingen te vinden in tabel 1.

Tabel 1.
Overzicht van afkortingen.

Afkorting	Betekenis
ASR	Automated Speech Recognition / automatische spraakherkenning
BLSTM	Bidirectional Long Short Term Memory / tweerichtings lange-kortetermijngeheugen

CNN	Convolutional Neural Network / convolutioneel neuraal netwerk
CPU	Central Processing Unit / centraal processorunit
DNN	Deep Neural Network / diep neuraal netwerk
FFNN	Feed Forward Neural Network / feedforward neuraal netwerk
GPU	Graphics Processing Unit / grafische processorunit
IEMOCAP	Interactive Emotional Dyandic Motion Capture
LSTM	Long Short Term Memory / lange- kortetermijngeheugen
MFCC	Mel Frequency Cepstral Coefficient / mel frequentiecepstrumcoëfficiënt
MFS	Mel Spectrogram Frequency / mel spectrogramfrequentie
NN	Neural Network / neuraal netwerk
RAVDESS	Ryerson Audio-Visual Database of Emotional Speech and Song
SER	Speech Emotion Recognition / spraakemotieherkenning
TDNN	Time Delay Neural Network / vertraagd neuraal netwerk
UA	Unweighted Accuracy / ongewogen nauwkeurigheid
WA	Weighted Accuracy / gewogen nauwkeurigheid

Emotie

Emotiecategorieën

In SER-onderzoek wordt meestal onderscheid gemaakt tussen de emotiecategorieën blij, verrast, boos, verdrietig, bang en neutraal (Marechal, 2019) om drie specifieke redenen die in relatie van het huidig onderzoek verklaard kunnen worden (Ekman, 1999):

1. Basisemoties verschillen van non-basisemoties in de mate van verschillende kenmerken.
2. Basisemoties verschillen van non-basisemoties in hun functie bij fundamentele taken in het dagelijks leven.
3. Basisemoties zijn in tegenstelling tot sommige non-basisemoties geen samenstelling van andere emoties.

De eerste reden van het verschil tussen basisemoties en non-basisemoties wordt duidelijk in de SER-experimenten die verder in het huidige onderzoek worden besproken. Deze

basisemoties kunnen inderdaad onderscheiden worden van non-basisemoties omdat het grote verschil tussen de emoties blij, verrast, boos, verdrietig, bang en neutraal ervoor zorgt dat het makkelijker wordt voor een SER-model om een emotie te herkennen. De tweede reden over het nut lijkt een vage term waarover men kan discussiëren, maar de derde reden is heel duidelijk en alle emoties die worden besproken in dit onderzoek voldoen aan deze voorwaarde.

Emotiekenmerken

Voor het huidig SER-onderzoek focussen we op emotie in het spraaksignaal waardoor representatie van emotie in andere vormen zoals gezichtsuitdrukkingen dus uitgesloten zijn. In deze sectie van het onderzoek wordt een overzicht gegeven van de manier waarop emotie in het spraaksignaal zit en hoe een spraaksignaal geanalyseerd kan worden om emotiekenmerken te herkennen.

Emotie in het spraaksignaal

Emotie komt in het spraaksignaal voor in de vorm van stemkwaliteit, toonhoogte, formantfrequenties en spraaktempo. In een onderzoek van Sim (2002) wordt beweerd dat emotie in het spraaksignaal het beste kan worden gemeten aan de hand van toonhoogte omdat dit element van een spraaksignaal altijd verandert wanneer de spreker van emotie verandert. Het is mogelijk om aan een verhoging, verlaging of een gebrek aan verandering te meten in welke emotionele staat iemand zich bevindt. Naast toonhoogte gebruikt dit onderzoek ook formanten om emotie te meten. Emotiekenmerken in het spraaksignaal van blij, boos en verrast werden beschreven. Per 0,05 seconden zijn de eerste en tweede formant (F1 en F2) bepaald om informatie over de emotie te verzamelen omdat F1 en F2 de dominante formanten zijn in het spraaksignaal (Sim, 2002).

Emotiekenmerken als verspreide informatie

Een probleem bij het analyseren van emotie in het spraaksignaal is het feit dat informatie van emotie heel verspreid ligt in de spraak. Om deze reden kan een spraakemotieherkenner niet aan een enkel kort stukje spraak (e.g. 10 ms) zien welke emotie in het signaal bevindt. Er is een analyse van een lang gedeelte van de spraak nodig. Het belang van het analyseren van emotie op een lange periode is in het onderzoek van Sarma (2018) besproken en bewezen. Een neuraal netwerk moet de mogelijkheid bieden om input data in meerdere kleine stukjes van bijvoorbeeld 10 ms spraak als input te nemen.

Korte introductie over neurale netwerken

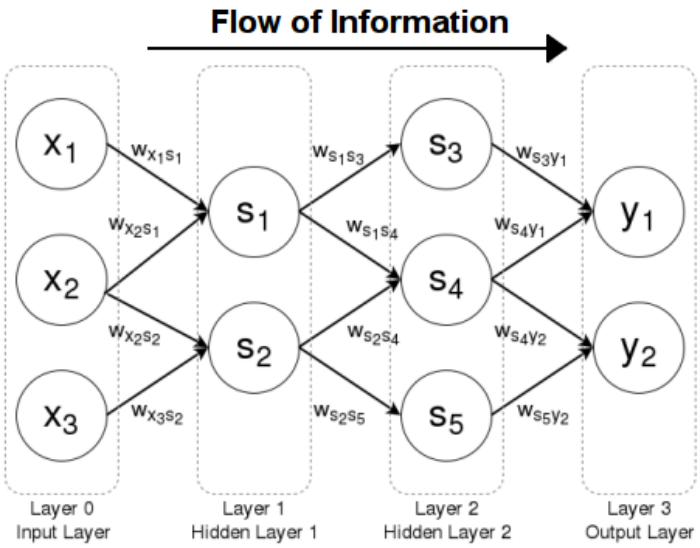
Geschiedenis

In de jaren tachtig zijn Deep Learning Neural Networks ontworpen, maar de techniek is recent pas geavanceerd genoeg om deze ontwerpen toe te passen. Voor het trainen van een DNN is namelijk een krachtige grafische processor unit (GPU) nodig omdat deze veel sneller matrix multiplicaties parallel kan uitvoeren vergeleken met een central processing unit (CPU) (Fatahalian, 2004). Het resultaat van deze DNN is dat de prestatie van deze zelflerende modellen enorm is toegenomen zoals duidelijk te zien is in de ImageNet competitie toen Krizhevsky et al. een Convolutional Neural Network (CNN) gebruikte om de winnende beeldherkenner te maken. Het AlexNet Model van Krizhevsky et al. bracht het percentage van fout herkende afbeeldingen van 26 procent (in 2011) naar 15 procent in 2012 (Shawahna, 2018). Na deze revolutie in 2012 werden Neurale Netwerken verder verbeterd tot het in bepaalde omstandigheden zelfs beter presteerde dan mensen (Alom, 2018). De reden dat de ontwikkelingen in automatische beeldherkenners relevant zijn voor spraakherkenning en spraakemotieherkenners, is omdat sommige technieken voor neurale netwerken op alle drie de vakgebieden kunnen worden toegepast zoals is aangetoond door Sainath (2015) en Kim

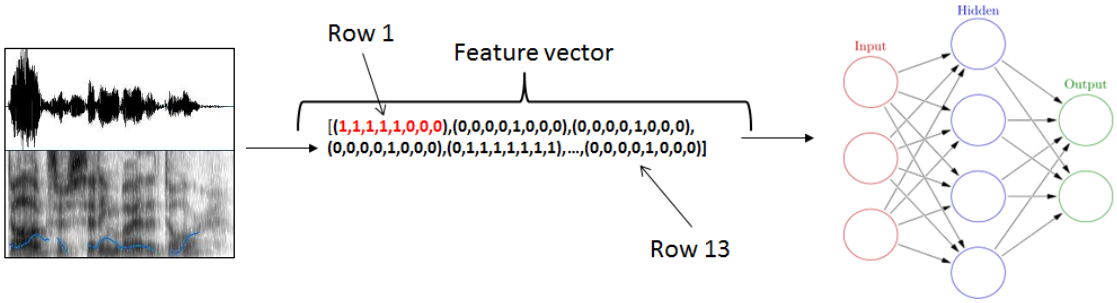
(2017) die Deep Temporal Neural Networks (DTNN) met identity skip connections toepasten op ASR en SER.

Spraaksignaalinformatie geleerd door een neuraal netwerk

Een neuraal netwerk werkt als volgt: Het netwerk bestaat op zijn minst uit twee lagen: een input laag (links in figuur 1) en een output laag (rechts in figuur 1). De lagen bestaan uit nodes die verbonden staan met aanliggende nodes doormiddel van weights ("w" in figuur 1). De informatie stroomt van links naar rechts in figuur 1. Eerst wordt er informatie gegeven aan de input laag. Een spraaksignaal moet eerst worden omgezet in vectoren zodat het de vorm heeft van getallen die aan de eerste laag nodes kan worden gegeven (figuur 2). De nodes in de input laag sturen informatie naar de volgende laag in het netwerk en de weight in de verbindingen bepalen hoe sterk de nodes in de volgende laag geactiveerd worden. Dit gaat door tot en met de output laag waar het netwerk een antwoord geeft op de vraag (e.g. "Welke emotie wordt er gerepresenteerd in het spraaksignaal?"). De vectoren in de output laag moeten dan worden omgerekend tot een emotie. Zodra het netwerk een voorspelling heeft gedaan, wordt het vergeleken met het juiste antwoord. Als de output dichtbij het juiste antwoord ligt, dan worden de weights van de verbindingen niet of nauwelijks gewijzigd, maar als de output ver van het juiste antwoord ligt, dan worden de weights van de verbindingen sterk gewijzigd.



Figuur 1. *Visuele representatie van een eenvoudige Feed Forward Neural Network.*



Figuur 2. *Feature vector van een spraaksignaal als input voor een neuraal netwerk*

De activatie van de nodes wordt bepaald door een combinatie van de activatiewaarde van de vorige laag nodes en de weights die de nodes met elkaar verbinden. Om uit te rekenen of en hoeveel een node wordt geactiveerd, wordt er een matrix multiplicatie toegepast. Het voorbeeldnetwerk in figuur 1 laat vier lagen van twee of drie nodes zien waardoor het berekenen van de activatiewaarden ook gemakkelijk (maar langzaam) door een mens gedaan zou kunnen worden, maar een netwerk die in staat is emotie te herkennen, bevat zoveel nodes en lagen dat er veel computerkracht nodig is om deze matrixmultiplicaties uit te rekenen. Een GPU is hier het meest geschikt voor omdat het parallelle berekeningen op hogere snelheid kan uitvoeren dan een CPU.

Backpropagation

Het aanpassen van de weights (beter bekend als ‘backpropagation’) van de verbindingen tussen de nodes gebeurt aan de hand van hoe goed of slecht het netwerk presteert in zijn voorspellingen. Een lage score zorgt voor grote aanpassingen en de een hoge score zorgt voor kleine aanpassingen. De ‘afstand’ tussen het voorspelde antwoord en het goede antwoord wordt gemeten, het netwerk wordt aangepast van de meest rechter laag naar de meest linker laag waarna het hele proces opnieuw begint met de volgende trainingsdata.

Testen

Nadat het netwerk getraind is, wordt het getest met data waar het model tot dan toe nog niet mee in contact is gekomen. Om deze reden wordt de beschikbare data voordat het trainen begint, opgedeeld in een trainingsset en testset. De score op de testdata wordt gemeten en gebruikt om te peilen hoe goed het netwerk presteert op een gegeven taak. Tijdens het testen wordt het model niet meer aangepast.

SER met neurale netwerken

Om het belang van neurale netwerken te benadrukken, kunnen we een voorbeeld analyseren van een toepassing van een neuraal netwerk op SER. In figuur 1 uit het onderzoeksverslag van Lee (2015) wordt SER in een neuraal netwerk gevisualiseerd. Het neuraal netwerk traint door heel vaak een spraaksignaal als input te nemen, een emotie als output te geven en die te vergelijken met de target emotie. Daarna wordt het neuraal netwerk getest door emotie van nieuwe spraaksignalen te voorspellen. Het neuraal netwerk model wordt beoordeeld op het aantal fout voorspelde emoties in de vorm van een Word Error Rate (WER). In het onderzoek van Lee (2015) wordt betekenis gegeven aan de verschillende lagen van het neuraal netwerk. Zoals eerder is uitgelegd, zijn deze nodes niets meer dan activatiewaarden die met elkaar verbonden zijn door weights. Lee (2015) doet een poging om betekenis te geven aan deze lagen. Elke laag bevat dus een combinatie van waarden die betekenis met zich meebrengen. Omdat het vaak heel moeilijk is om te zien wat een neuraal

netwerk precies doet, is het belangrijk om de waarde van SER te laten zien doormiddel van uitleg van deze lagen.

State-of-the-art SER

De huidige state-of-the-art spraakherkenners zijn gemaakt met Kaldi (Povey 2011) of HTK (Young, 1993). Voor dit experiment nemen we een bestaand state-of-the-art SER-model van Sarma (2018, 2019) als baseline en vergelijken de data, parameters en architectuur met een eigen SER-model in Python.

PyTorch-Kaldi

Kaldi is een open-source software en wordt gebruikt om state-of-the-art spraakherkenners te maken. De PyTorch library is een verzameling van functies en methoden in de programmeertaal Python dat gebruikt wordt voor deep learning projecten met neurale netwerken. Het PyTorch-Kaldi project is gemaakt om PyTorch toe te passen op Kaldi. Het is ontwikkeld door (Ravanelli, 2019) zorgt ervoor dat feature extraction, label/alignment computation, en decoding kunnen worden toegepast op akoestische modellen. Dezelfde technieken van deze toolkit zouden gebruikt kunnen worden om emotie te herkennen.

Omdat het PyTorch-Kaldi toolkit gebruik maakt van Python, is het relevant om onderzoek te doen naar de mogelijkheden van SER in Python. In dit onderzoek wordt eerst een overzicht gegeven van de emotiekenmerken en technieken om deze te herkennen met een DNN. Daarna worden deze emotiekenmerken geïmplementeerd in een spraakemotieherkenner in Python. Omdat Python/Pytorch en Kaldi zijn gekoppeld met het Pytorch-Kaldi toolkit, kunnen de conclusies van dit onderzoek over wat welk en niet werkt in Python, worden toegepast in een state-of-the-art SER in Kaldi.

SER in Kaldi

Kaldi is een open-source toolkit die kan worden gebruikt voor SER door training- en testdata te verdelen in mapjes die je definieert met paths in een script. In datzelfde script bereid je de data voor door de opnames gelijk te leggen aan de emoties en daarna één of meerdere training methode(s) aan te roepen (zie appendix A voor voorbeeldscript). Een recent SER-onderzoek in Kaldi werd verricht in 2018 en 2019 door Sarma. Spraakopnames van de Interactive Emotional Dyandic Motion Capture (IEMOCAP) database werden in Kaldi gebruikt om verschillende architecturen voor SER te testen. In dit onderzoek werden meerdere DNNs vergeleken op hun vermogen om emotiekenmerken over lange termijn te herkennen. De TDNN met Long Short Term Memory (LSTM) bleek het meest geschikt te zijn voor het analyseren van emotiekenmerken over lange tijdsperiodes.

Trainingsdata

Data is een belangrijk component van het ontwerp van een model omdat de informatie in de trainingsdata bepaalt hoe de verbindingen in het model eindigen voordat het getest wordt op de vaardigheid in het voorspellen van emotie uit een spraaksignaal. De input die in het onderzoek van Sarma aan het neuraal netwerk wordt gegeven bestaat uit frames van 40 ms lang.

Vier emotiecategorieën uit de IEMOCAP database werden in dit onderzoek van elkaar onderscheiden: neutraal, boos, verdrietig en blij. De data die gebruikt is voor dit experiment bestond uit 12 uur aan audiovisuele data van vijf vrouwelijke en vijf mannelijke acteurs. De data wordt opgedeeld in 75% trainingsdata en 25% testdata.

Emotie verspreidt over lange tijd

Het lastige van SER is dat je het niet kan doen over een korte tijdsperiode. Omdat emotiekenmerken zijn verspreid over een lange termijn, moet de architectuur daarop zijn aangepast. Het ontwerp van het neuraal netwerk kan hierop worden aangepast door het op een langere tijdsperiode afhankelijk te maken.

Het probleem van verspreide emotie in het spraaksignaal is dat een spraakemotieherkenner niet aan slechts één kort fragment van een paar milliseconden kan zien welke emotie er aanwezig is. Er is een analyse van een lang gedeelte van de spraak nodig. Om emotiecategorieën te representeren op een manier dat alle informatie van de emotie geanalyseerd kan worden door het neuraal netwerk, wordt er in het onderzoek van Pandey (2019) aangeraden om gebruikt te maken van een Log-Mel Spectrogram in combinatie met CNN met een Bidirectional Long Short Term memory (BLSTM). Het voordeel van deze architectuur is dat het informatie van de emotiekenmerken kan vasthouden en een voorspelling kan doen van de emotie in het spraaksignaal op basis van een langere tijdsperiode dan een ander architectuur kan bieden.

Testen van prestatie

De prestatie van de spraakemotieherkenner zijn op twee manieren weergegeven: weighted accuracy (WA) en unweighted accuracy (UA). WA staat voor de algemene classificatienauwkeurigheid of de nauwkeurigheid in het voorspellen van de emotie. UA staat voor de gemiddelde aantal onthouden emotiecategorieën. UA geeft een beter beeld geeft van de prestatie dan WA wanneer er een onbalans is tussen emotieklassen in termen van hoeveelheid beschikbare data.

Het vergroten van de dataset om een neuraal netwerk mee te trainen en te testen heeft meestal positieve gevolgen. Hier werd datavergroting toegepast doormiddel van data perturbatie (Ko, 2015). Bij data perturbatie worden de spraaksignalen vermenigvuldigd naar drie varianten waarvan de amplitudes vermenigvuldigd worden met 1.1, 1.0 en 0.9. Het gevaarlijke van data augmentatie is dat overfitting gemakkelijk kan plaatsvinden. Overfitting gebeurt wanneer het neuraal netwerk generalisaties maakt die er eigenlijk niet zijn waardoor het weinig fouten zal maken in de trainingsdata, maar relatief veel fouten zal maken bij de testdata. De resultaten van Sarma laten zien dat hun methode van dataperturbatie op SER

effectief is en de nauwkeurigheidsscore significant verhoogt van 60.27% naar 66.07% WA en van 48.84% naar 57.21% UA.

Na het bovenstaande onderzoek kwam er een vervolg (Sarma, 2019) waarin verklaard werd waarom een TDNN in combinatie met een LSTM helpt bij het verbeteren van SER. Met de aanvulling van deze input verlaagde de foute voorspellingen van LSTM met 3.42% en TDNN-LSTM met 16.48%.

SER in python

In dit experiment worden de eigenschappen van de state-of-the-art SER van Sarma (2019) gebruikt als baseline voor het testen van verschillende parameters en architecturen in een SER in Python**.** Het ontwerp voor het nieuwe SER model is gebaseerd op het Python SER project van de website data-flair.training (Appendix B). Omdat de SER van Sarma (2019) een andere database, architectuur en programma gebruikt, is het lastig om een vergelijking te maken, maar binnen het Python experiment kunnen bepaalde datasets, parameters en architecturen met elkaar vergeleken worden om te zien wat wel en niet werkt voor een SER. Met de informatie over de datasets, architectuur en parameters die een positief of negatief effect hebben op de nauwkeurigheidsscore van een SER, kan er een conclusie worden gemaakt over wat wel en niet zou werken in de state-of-the-art SER van Sarma (2019).

Data

Dit Python project maakt gebruik van de libraries Librosa, Soundfile, Sklearn en Numpy en de modules OS, Glob en Pickle. De data is een selectie van de Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)-dataset en bestaat uit 60 verschillende opnames van zinnen in 8 verschillende emoties ingesproken door 24 acteurs (in totaal 1440 opnames). Het script volgt een traditionele manier van spraakherkenning waarbij het spraaksignaal voor analyse wordt omgezet in een spectrogram op een Mel schaal.

Emotiecategorieën

De RAVDESS-database bevat de emoties neutraal, kalm, blij, verdrietig, boos, bang, walging en verrast. In het Python script kunnen deze categorieën gemakkelijk worden geselecteerd in het Python script. Dit heeft er echter tot gevolg dat alleen de opnames van de geselecteerde emotie gebruikt worden waardoor de trainings- en testdata verminderd.

Neuraal netwerkmodel

Het neuraal netwerk die in dit Python script wordt aangeroepen is een FFN met een Multi Layer Perceptron (MLP) Classifier van Sklearn. Voor dit experiment zijn de voordelen van een MLP de mogelijkheid tot het leren van non-lineaire modellen zoals bij SER het geval is. Het nadeel van de MLP Classifier van Sklearn is dat het geen GPU ondersteunt, maar dat maakt de uitvoering van dit experiment veel toegankelijker vergeleken met andere neurale netwerken omdat het minder geavanceerde techniek vereist. De architectuur, parameters en attributen zijn weergegeven in appendix C, D en E.

Data verdeling

In dit experiment wordt een neuraal netwerk getraind en getest op geselecteerde data van de RAVDESS database. 80% van de 1440 opnames worden gebruikt voor de trainingsdata en 20% wordt apart gehouden totdat het model wordt getest op prestatie van het herkennen van emotie.

Emotiekenmerken

Bij het analyseren van het geluidssignaal in de opnames let het model op drie kenmerken bij het herkennen van de emotie: Mel Frequency Cepstral Coefficient (MFCC), toonhoogte/chroma en Mel Spectrogram Frequency (MSF).

Training

De MLP Classifier model bevat een feed forward neuraal netwerk (FFNN) die wordt getraind op 75% van de opnames. De emotie van elke opname is aangegeven in het derde getal van de naam. ‘01’ staat voor de emotie ‘neutraal’ en een voorbeeldnaam van een

opname met neutrale spraak is bijvoorbeeld ’03-01-01-02-01-01-09’. Tijdens het trainen voorspelt het model het derde getal in de naam op basis van de drie kenmerken: MFCC, chroma en MSF. Het model krijgt tijdens het trainen feedback door het derde getal van de naam te vergelijken met de voorspelde emotie. Dus wanneer het model bijvoorbeeld de emotie ‘neutraal’ heeft herkend en het derde getal van de naam van de opname is niet ‘01’, dan worden de verbindingen tussen de neuronen tijdens de backpropagatie sterk aangepast om de volgende herkenning te verbeteren. Heeft het model wel een correcte voorspelling, dan worden er geen aanpassingen gedaan aan de verbindingen tussen de neuronen.

Testen

Tijdens het testen wordt het model beoordeel op de nauwkeurigheid in zijn vaardigheid om emotie correct te herkennen. Het herkennen van een emotie is in dit experiment niet meer dan een voorspelling van het derde getal in de naam op basis van de kenmerken die het model analyseert met ervaring die tijdens het trainen is opgedaan. Dus als het model vaak een lage monotone klank heeft gehoord in combinatie met een ‘01’ als derde getal in de naam, dan zal het model deze ervaring gebruiken om een ‘01’ als derde getal in de naam te voorspellen zodra het een lage monotone klank hoort.

Resultaten

Het model werd getest op 20% van de opnames en de score is weergegeven als een nauwkeurigheidspercentage (hoger is beter) in tabel 2 en 3. Het nauwkeurigheidspercentage is berekend door de accuracy_score() functie uit Sklearn en is afgerond op twee decimalen. De score op de voorspelling van emotie tussen alle emoties in het script is 51.39%. Na deze complete training en test is het model getraind en getest op combinaties van emoties om een klein inzicht te geven welke emoties makkelijker of moeilijk van elkaar te onderscheiden zijn en hoe dezen bij elkaar de score van 51.39% tot stand brengen. In tabel 2 is te zien dat de emotie ‘kalmte’ door het model makkelijk te onderscheiden is van met name ‘blij’, ‘boos,

‘bang en ‘verrast’. Uit deze scores kan men opmaken dat het verschil tussen deze emoties in toonhoogte, MFFC, en MSF makkelijk te herkennen is door het model met deze trainingsdata, architectuur en parameters. De emoties ‘kalm’ en ‘neutraal’, ‘neutraal’ en ‘verrast’, ‘bang’ en blij’, en ‘verrast’ en ‘walging’ zijn de vier combinaties waar het model de meeste moeite mee heeft. De verschillen in het spraaksignaal van de trainingsdata zijn niet geschikt voor het neurale netwerk om generalisaties van te maken die hoge scores opleveren met de testdata.

Tabel 2.
Scores van combinaties van alle emoties uit de RAVDESS-database.

	Neutraal	Kalm	Blij	Verdrietig	Boos	Bang	Walging	Verrast
Neutraal	-							
Kalm	75.00	-						
Blij	84.72	95.83	-					
Verdrietig	81.94	85.42	78.12	-				
Boos	80.56	97.71	88.54	89.58	-			
Bang	90.28	96.88	75.00	86.46	86.46	-		
Walging	80.56	86.46	88.54	87.50	81.25	85.42	-	
Verrast	56.94	98.96	81.25	88.54	89.58	91.67	71.88	-

Tabel 3.
Nauwkeurigheidsscores in percentage wanneer er steeds één emotie wordt weggelaten.

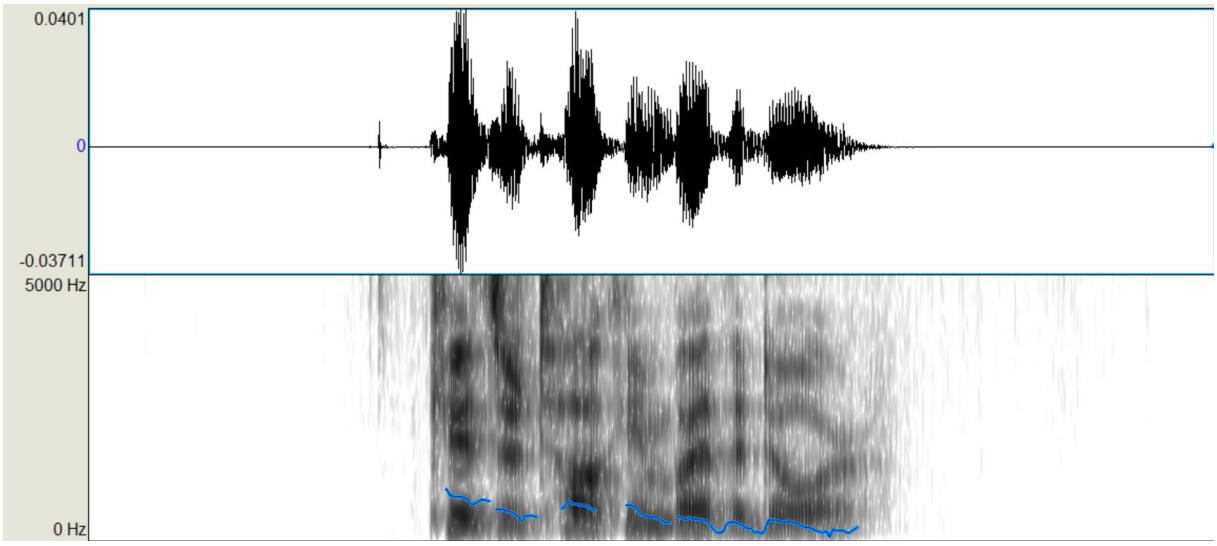
Emoties	Nauwkeurigheid
‘neutraal', 'kalm', ‘blij’, ‘verdrietig’, ‘boos’, ‘bang’, ‘walging’	52.40
‘neutraal', 'kalm', ‘blij’, ‘verdrietig’, ‘boos’, ‘bang’, ‘verrast’	65,90
‘neutraal', 'kalm', ‘blij’, ‘verdrietig’, ‘boos’, ‘walging', ‘verrast’	57.23
‘neutraal', 'kalm', ‘blij’, ‘verdrietig’, ‘bang’, ‘walging', ‘verrast’	56.40
‘neutraal', 'kalm', ‘blij’, ‘boos’, ‘bang’, ‘walging', ‘verrast’	45.60
‘neutraal', 'kalm', ‘verdrietig’, ‘boos’, ‘bang’, ‘walging', ‘verrast’	52.00
‘neutraal', ‘blij’, ‘verdrietig’, ‘boos’, ‘bang’, ‘walging', ‘verrast’	51.20
'kalm', ‘blij’, ‘verdrietig’, ‘boos’, ‘bang’, ‘walging', ‘verrast’	50.40
'kalm', ‘blij’, ‘verdrietig’, ‘boos’, ‘bang’, ‘verrast’	73.38

Tijdens het experiment zijn combinaties van twee en zeven emoties getest en er kwam een patroon uit voort waarin de emoties 'kalm', ‘blij’, ‘verdrietig’, ‘boos’, ‘bang’, ‘verrast’ de hoogste nauwkeurigheidsscores opleverden. Er zijn ook verschillende parameters getest en de meest geschikte parameters voor een FFNN met een MLP Classifier zijn weergegeven in Appendix C, D en E. Er zijn architecturen die geschikter zijn voor SER, maar vanwege

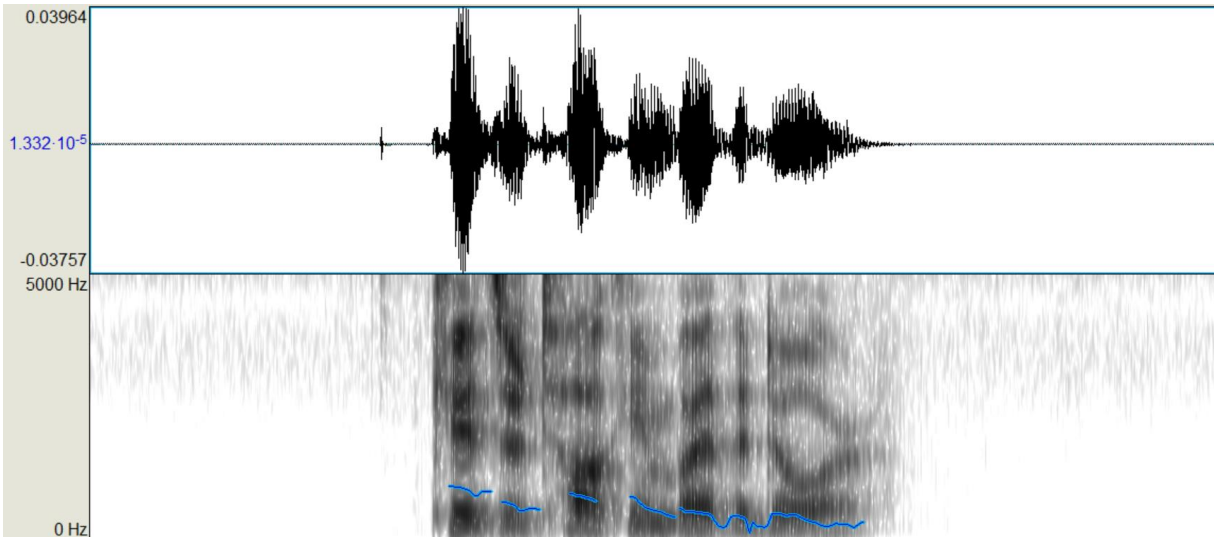
technische limieten kan er in dit onderzoek geen neuraal netwerk worden getest met een geavanceerdere architectuur zoals een LSTM RNN.

Dataperturbatie in Python

Voor neurale netwerken geld dat meer training gelijk staat aan hogere nauwkeurigheid Het is echter lastig om aan veel trainingsdata te komen waardoor veel data wordt herhaald tijdens het trainen. Helaas kan teveel herhaling zorgen voor een grotere kans op overfitting. In Ko (2015) en Sarma (2018) snelheid perturbatie toegepast om overfitting te voorkomen. Snelheid perturbatie is het kopiëren van data (figuur 3) waarbij die kopieën vervolgens worden versneld of vertraagd (figuur 4). De opnames van de RAVDESS database zijn verdubbeld en versneld met een factor van 1.1. De nauwkeurigheid van een FFNN getraind op de originele en versnelde data en getest op de emoties 'kalm', ‘blij’, ‘verdrietig’, ‘boos’, ‘bang’ en ‘verrast’ is gestegen naar 77,60%. Dit is een stijging van 4,22% vergeleken met de FFNN zonder snelheid perturbatie.



Figuur 3. *Neutrale emotie op 100% snelheid.*



Figuur 4. *Neutrale emotie op 110% snelheid.*

Met het verhogen van de snelheid veranderd niet alleen de lengte van de opname en de toonhoogte, maar ook de toonhoogtecontour zoals te zien is in figuur 3 en 4. Dit zorgt ervoor dat het neurale netwerk spraakpatronen in de versnelde opname herkent die anders zijn dan de originele opname, maar toch kenmerkend zijn voor de betreffende emotie. De emotiekenmerken gaan dus niet verloren terwijl het neurale netwerk meer data heeft om mee te leren.

Advies voor SER in Kaldi

Stap 1: Data

De spraaksignalen van de volgende emotiecategorieën zijn zeer geschikt voor een neuraal netwerk om onderscheid tussen te maken: ‘kalm’, ‘blij’, ‘boos’, ‘bang’ ‘verdrietig’ en ‘verrast’. In voorgaand onderzoek werden er meestal onderscheid gemaakt tussen zes verschillende emotiecategorieën. Meer categorieën zou leiden tot lagere nauwkeurigheid, terwijl minden categorieën leidt tot een vermindering in toepasbaarheid in de praktijk. De formanten en toonhoogte zijn op een Mel schaal goed van elkaar te onderscheiden omdat het neuraal netwerk generalisaties kan maken op basis van de trainingsdata van de IEMOCAP- en

RAVDESS-database en omdat een SER de hoogste score bereikt met deze combinatie van emotiecategorieën op de testdata.

De data moet worden opgesplitst in een trainingset en testset zodat het neuraal netwerk getest kan worden op spraakopnames die het nooit eerder is tegengekomen.

Snelheid perturbatie is een geschikte methode om ervoor te zorgen dat het neurale netwerk meer data heeft om van te leren terwijl de emotiekenmerken behouden blijven en terwijl overfitting voorkomen wordt. Het is aan te raden om naast snelheid perturbatie zoveel mogelijk data te gebruiken zonder dat er overfitting plaatsvindt.

Stap 2: Algoritme

De huidige state-of-the-art SER gebruiken geen FFNN, maar een TDNN LSTM omdat emotiekenmerken verspreid zitten over een lange tijdsperiode van het spraaksignaal en omdat TDNN LSTM modellen zijn bewezen in state-of-the-art SER-modellen (Sarma, 2018, 2019). Daarnaast is het belangrijk dat de juiste emotiekenmerken geanalyseerd kunnen worden. Om emotiecategorieën te representeren op een manier dat alle informatie van de emotie geanalyseerd kan worden door het neuraal netwerk, raad ik aan om in Kaldi een trainingsmethode aan te roepen die rekening houdt met een Log-Mel Spectrogram.

Stap 3: Testen

Bij het testen van een neuraal netwerk is het belangrijk om een testset te gebruiken waar het tot dan toe nooit eerder aan is blootgesteld. De testset moet uiteraard bestaan uit dezelfde emoties waarop het model is getraind en de scores kunnen worden weergegeven in zowel weighted als unweighted accuracy. UA krijgt voorkeur boven WA omdat bij het trainen van een neuraal netwerk je altijd zoveel mogelijk data wilt hebben. Het nadeel hiervan is dat meer data een onbalans kan creëren tussen de klassen in termen van hoeveelheid data en daarmee de nauwkeurigheid tijdens het testen kan beïnvloeden, maar dit kan worden opgelost door de scores weer te geven als WA.

Referenties

Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., ... & Asari, V. K. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*.

Clore, G. L., & Huntsinger, J. R. (2007). How emotions inform judgment and regulate thought. *Trends. cognitive sciences*, *11*(9), 393-399.

Davletcharova, A., Sugathan, S., Abraham, B., & James, A. P. (2015). Detection and analysis of emotion from speech signals. *arXiv preprint arXiv:1506.06832*.

Ekman, P. (1999). Basic emotions. *Handbook of cognition and emotion*, *98*(45-60), 16.

Fatahalian, K., Sugerman, J., & Hanrahan, P. (2004, August). Understanding the efficiency of GPU algorithms for matrix-matrix multiplication. *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (pp. 133-137).

Grimm, M., Kroschel, K., Harris, H., Nass, C., Schuller, B., Rigoll, G., & Moosmayr, T. (2007, September). On the necessity and feasibility of detecting a driver's emotional state while driving. *International Conference on Affective Computing and Intelligent Interaction* (pp. 126-138). Springer, Berlin, Heidelberg.

Hossain, M. S., & Muhammad, G. (2015). Cloud-assisted speech and face recognition framework for health monitoring. *Mobile Networks and Applications*, *20*(3), 391-399.

Kamaruddin, N., & Wahab, A. (2011, October). Heterogeneous driver behavior state recognition using speech signal. *Proceedings of the 10th WSEAS international conference on System science*

Kim, J., Englebienne, G., Truong, K. P., & Evers, V. (2017, October). Deep temporal models using identity skip-connections for speech emotion recognition. *Proceedings of the 25th ACM international conference on Multimedia* (pp. 1006-1013).

Ko, T., Peddinti, V., Povey, D., & Khudanpur, S. (2015). Audio augmentation for speech

recognition. *Sixteenth Annual Conference of the International Speech Communication Association. and simulation in engineering* (pp. 207-212).

Lee, F. M., Li, L. H., & Huang, R. Y. (2008, February). Recognizing low/high anger in speech for call centers. *International Conference on Signal Processing, Robotics and utomation* (pp. 171-176).

Lee, J., & Tashev, I. (2015). High-level feature representation using recurrent neural network for speech emotion recognition. *Sixteenth annual conference of the international speech communication association*.

Morrison, D., Wang, R., & De Silva, L. C. (2007). Ensemble methods for spoken emotion recognition in call-centres. *Speech communication, 49*(2), 98-112.

Marechal, C., Mikołajewski, D., Tyburek, K., Prokopowicz, P., Bougueroua, L., Ancourt, C., & Węgrzyn-Wolska, K. (2019). Survey on AI-Based Multimodal Methods for Emotion Detection. *High-Performance Modelling and Simulation for Big Data Applications* (pp. 307-324). Springer, Cham.

Nass, C., Jonsson, I. M., Harris, H., Reaves, B., Endo, J., Brave, S., & Takayama, L. (2005, April). Improving automotive safety by pairing driver emotion and car voice emotion. *CHI'05 extended abstracts on Human factors in computing systems* (pp. 1973-1976).

O'Shaughnessy, D. (2008). Automatic speech recognition: History, methods and challenges. *Pattern Recognition, 41*(10), 2965-2979.

Pandey, S. K., Shekhawat, H. S., & Prasanna, S. R. M. (2019, April). Deep Learning Techniques for Speech Emotion Recognition: A Review. *2019 29th International Conference Radioelektronika (RADIOELEKTRONIKA)* (pp. 1-6). IEEE.

Petrushin, V. (1999, November). Emotion in speech: Recognition and application to call centers. *Proceedings of artificial neural networks in engineering* (Vol. 710, p. 22).

Pieraccini, R. (2012). From AUDREY to Siri: Is speech recognition a solved

problem?. *International Computer Science Institute at Berkeley.*

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., ... & Silovsky, J. (2011). The Kaldi speech recognition toolkit. *IEEE 2011 workshop on automatic speech recognition and understanding* (No. CONF). IEEE Signal Processing Society.

Ravanelli, M., Parcollet, T., & Bengio, Y. (2019, May). The pytorch-kaldi speech recognition toolkit. *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6465-6469). IEEE.

Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015, April). Convolutional, long short-term memory, fully connected deep neural networks. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4580-4584). IEEE.

Sarma, M., Ghahremani, P., Povey, D., Goel, N. K., Sarma, K. K., & Dehak, N. (2018, September). Emotion Identification from Raw Speech Signals Using DNNs. *Interspeech* (pp. 3097-3101).

Sarma, M., Ghahremani, P., Povey, D., Goel, N. K., Sarma, K. K., & Dehak, N. (2019). Improving Emotion Identification Using Phone Posteriors in Raw Speech Waveform Based DNN. *INTERSPEECH* (pp. 3925-3929).

Shawahna, A., Sait, S. M., & El-Maleh, A. (2018). FPGA-based accelerators of deep learning networks for learning and classification: A review. *IEEE Access*, *7*, 7823-7859.

Sim, K. B., Park, C. H., Lee, D. W., & Joo, Y. H. (2002). Emotion recognition based on frequency analysis of speech signal. *International Journal of Fuzzy Logic and Intelligent Systems*, *2*(2), 122-126.

Szwoch, M., & Szwoch, W. (2015). Emotion recognition for affect aware video games. *Image Processing & Communications Challenges 6* (pp. 227-236). Springer, Cham.

Tickle, A., Raghu, S., & Elshaw, M. (2013). Emotional recognition from the speech signal for

a virtual education agent. *Journal of Physics: Conference Series* (Vol. 450, No. 1, p. 012053). IOP Publishing.

Torous, J., Friedman, R., & Keshavan, M. (2014). Smartphone ownership and interest in mobile applications to monitor symptoms of mental health conditions. *JMIR mHealth and uHealth*, *2*(1), e2.

Young, S. J., & Young, S. (1993). The HTK hidden Markov model toolkit: Design and philosophy.

Appendix A. Voorbeeldscript SER in Kaldi

```
#!/bin/bash
. ./path.sh || exit 1
. ./cmd.sh || exit 1
nj=1 # number of parallel jobs - 1 is perfect for such a small dataset
lm_order=1 # language model order (n-gram quantity) - 1 is enough for digits grammar
# Safety mechanism (possible running this script with modified arguments)
. utils/parse_options.sh || exit 1
[[ $# -ge 1 ]] && { echo "Wrong arguments!"; exit 1; }
# Removing previously created data (from last run.sh execution)
rm -rf exp mfcc data/train/spk2utt data/train/cmvn.scp data/train/feats.scp data/train/split1
        data/test/spk2utt data/test/cmvn.scp data/test/feats.scp data/test/split1 data/local/lang data/lang
        data/local/tmp data/local/dict/lexiconp.txt
echo
echo "===== PREPARING ACOUSTIC DATA ====="
echo
# Needs to be prepared by hand (or using self written scripts):
#
# spk2gender [<speaker-id> <gender>]
# wav.scp [<uterranceID> <full_path_to_audio_file>]
# text [<uterranceID> <text_transcription>]
# utt2spk [<uterranceID> <speakerID>]
# corpus.txt [<text_transcription>]
# Making spk2utt files
utils/utt2spk_to_spk2utt.pl data/train/utt2spk > data/train/spk2utt
utils/utt2spk_to_spk2utt.pl data/test/utt2spk > data/test/spk2utt
echo
echo "===== FEATURES EXTRACTION ====="
echo
# Making feats.scp files
mfccdir=mfcc
# Uncomment and modify arguments in scripts below if you have any problems with data sorting
# utils/validate_data_dir.sh data/train # script for checking prepared data - here: for data/train
        directory
# utils/fix_data_dir.sh data/train # tool for data proper sorting if needed - here: for data/train directory
steps/make_mfcc.sh --nj $nj --cmd "$train_cmd" data/train exp/make_mfcc/train $mfccdir
steps/make_mfcc.sh --nj $nj --cmd "$train_cmd" data/test exp/make_mfcc/test $mfccdir
# Making cmvn.scp files
steps/compute_cmvn_stats.sh data/train exp/make_mfcc/train $mfccdir
steps/compute_cmvn_stats.sh data/test exp/make_mfcc/test $mfccdir
echo
echo "===== PREPARING LANGUAGE DATA ====="
echo
# Needs to be prepared by hand (or using self written scripts):
#
# lexicon.txt [<word> <phone 1> <phone 2> ...]
# nonsilence_phones.txt [<phone>]
# silence_phones.txt [<phone>]
```

```
# optional_silence.txt [<phone>]
# Preparing language data
utils/prepare_lang.sh data/local/dict "<UNK>" data/local/lang data/lang
echo
echo "===== LANGUAGE MODEL CREATION ====="
echo "===== MAKING lm.arpa ====="
echo
loc=`which ngram-count`;
if [ -z $loc ]; then
if uname -a | grep 64 >/dev/null; then
sdir=$KALDI_ROOT/tools/srilm/bin/i686-m64
else
sdir=$KALDI_ROOT/tools/srilm/bin/i686
fi
if [ -f $sdir/ngram-count ]; then
echo "Using SRILM language modelling tool from $sdir"
export PATH=$PATH:$sdir
else
echo "SRILM toolkit is probably not installed.
Instructions: tools/install_srilm.sh"
exit 1
fi
fi
local=data/local
mkdir $local/tmp
ngram-count -order $lm_order -write-vocab $local/tmp/vocab-full.txt -wbdiscount -text
        $local/corpus.txt -lm $local/tmp/lm.arpa
echo
echo "===== MAKING G.fst ====="
echo
lang=data/lang
arpa2fst --disambig-symbol=#0 --read-symbol-table=$lang/words.txt $local/tmp/lm.arpa $lang/G.fst
echo
echo "===== MONO TRAINING ====="
echo
steps/train_mono.sh --nj $nj --cmd "$train_cmd" data/train data/lang exp/mono || exit 1
echo
echo "===== MONO DECODING ====="
echo
utils/mkgraph.sh --mono data/lang exp/mono exp/mono/graph || exit 1
steps/decode.sh --config conf/decode.config --nj $nj --cmd "$decode_cmd" exp/mono/graph data/test
        exp/mono/decode
echo
echo "===== MONO ALIGNMENT ====="
echo
steps/align_si.sh --nj $nj --cmd "$train_cmd" data/train data/lang exp/mono exp/mono_ali || exit 1
echo
echo "===== TRI1 (first triphone pass) TRAINING ====="
echo
```

```
steps/train_deltas.sh --cmd "$train_cmd" 2000 11000 data/train data/lang exp/mono_ali exp/tri1 || exit
    1
echo
echo "===== TRI1 (first triphone pass) DECODING ====="
echo
utils/mkgraph.sh data/lang exp/tri1 exp/tri1/graph || exit 1
steps/decode.sh --config conf/decode.config --nj $nj --cmd "$decode_cmd" exp/tri1/graph data/test
    exp/tri1/decode
echo
echo "===== run.sh script is finished ====="
echo
```

Appendix B. SER script in Python

```
# Bachelor eindwerkstuk (scriptie)
# Universiteit Utrecht
# Rens Kersbergen

# This script was based on https://data-flair.training/blogs/python-mini-project-speech-emotion-recognition/
# and altered to extract the highest score with all emotions in the RAVDESS database
# A sample of the RAVDESS data set was used to train and test the model.

import librosa
import soundfile
import os, glob, pickle
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

# Extract features (mfcc, chroma, mel) from a sound file
def extract_feature(file_name, mfcc, chroma, mel):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        if chroma:
            stft=np.abs(librosa.stft(X))
        result=np.array([])
        if mfcc:
            mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axis=0)
            result=np.hstack((result, mfccs))
        if chroma:
            chroma=np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
            result=np.hstack((result, chroma))
        if mel:
            mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
            result=np.hstack((result, mel))
        return result

# Emotions in the RAVDESS dataset
emotions={
  '01':'neutral',
  '02':'calm',
  '03':'happy',
  '04':'sad',
  '05':'angry',
  '06':'fearful',
  '07':'disgust',
```

```
  '08':'surprised'
}

# Emotions to observe
observed_emotions=['neutral', 'calm', 'happy', 'sad','angry', 'fearful', 'disgust', 'surprised']

# Load the data and extract features for each sound file
def load_data(test_size=0.2):
    x,y=[],[]
    for file in glob.glob("C:\\Users\\Rens
Kersbergen\\Documents\\Taalwetenschap\\Scriptie\\SER in Python\\Actor_*\\*.wav"):
        file_name=os.path.basename(file)
        emotion=emotions[file_name.split("-")[2]]
        if emotion not in observed_emotions:
            continue
        feature=extract_feature(file, mfcc=True, chroma=True, mel=True)
        x.append(feature)
        y.append(emotion)
    return train_test_split(np.array(x), y, test_size=test_size, random_state=9)


# Split the dataset
x_train,x_test,y_train,y_test=load_data(test_size=0.2)

# Get the shape of the training and testing datasets
print((x_train.shape[0], x_test.shape[0]))

# Get the number of features extracted
print(f'Features extracted: {x_train.shape[1]}')

# Initialize the Multi Layer Perceptron Classifier
model=MLPClassifier(alpha=0.01, batch_size=256, epsilon=1e-08,
hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=500, solver='adam',
learning_rate_init=0.02)

# Train the model
model.fit(x_train,y_train)

# Predict for the test set
y_pred=model.predict(x_test)

# Calculate the accuracy of our model
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
# Print the accuracy
print("Accuracy: {:.2f}%".format(accuracy*100))
```

Appendix C. Non-default parameters van het neuraal netwerk

Alpha = 0.01
Batch_size = 250
Epsilon = 1e-8
Hidden_layer_sizes = 300
Max_iter = 500
Test_size=0.2

Appendix D. Default parameters van het neuraal netwerk

Activation = ‘relu’
Solver = ‘adam’
Learning_rate = ‘constant’
Learning_rate_init = 0.001
Power_t = 0.5
Shuffle = True
Random_state = None
Tol = 1e-4
Verbose = False
Warm_start = False
Momentum = 0.9
Nesterovs_momentum = True
Early stopping = False
Validatoin_fraction = 0.1
Beta_1 = 0.9
Beta_2 = 0.999
N_iter_no_change = 10
Max_fun = 15000

Appendix E. Attributen van het neuraal netwerk

Classes_ = ndarray
Loss_ = float
Coefs_ = list, length n_layers -1
Intercepts_ = list, length n_layers -1
N_iter_ = int
N_layers = int
N_outputs_ = int
Out_activation_ = string

Appendix F**. Logboek scriptie**

23-04 12:00-17:00
Doel voor de eerste twee weken: Kaldi installeren, htk installeren
Ik heb Kaldi succesvol geinstalleerd op windows 10. Voordat ik ga beginnen het invoeren van audio files, wil ik uitzoeken of Kaldi wel geschikt is voor mij en mijn laptop. Ik vermoed dat Kaldi beter werkt op Linux (in een virtual machine). Spraakherkenning in Python is blijkbaar ook mogelijk en is voor mij het meest overzichtelijk https://data-flair.training/blogs/Python-mini-project-speech-emotion-recognition/
Volgende stap: emotieherkenning in Python maken en proberen kleine aanpassingen te maken. Hier heb ik weinig tijd voor nodig en als het werkt, kan dit mij helpen om mijn deelvragen te beantwoorden.

24-04 10:00-11:00
SER in Python gemaakt aan de hand van https://data-flair.training/blogs/python-mini-project-speech-emotion-recognition/ . Er worden 1440 audiobestanden (24 acteurs met ieder 60 bestanden) gebruikt om 8 emoties te herkennen. Dit is een sample van de RAVDESS dataset (Livingstone, R. 2018) Dit script kan ik gebruiken om experimenten te doen met het herkennen van verschillende emoties. Die informatie kan ik gebruiken om toe te passen in Kaldi of HTK. De volledige RAVDESS dataset kan ik gebruiken als ik meer data nodig heb.
Volgende stap: RAVDESS steeksproefgewijs beluisteren en kwaliteit beschrijven. Daarna RAVDESS gebruiken in Kaldi.

11:00-17:00
Gelezen en een korte samenvatting gemaakt zodat ik een overzicht heb en kan zien wat ik nodig heb:
http://ocean.kisti.re.kr/downfile/volume/kfis/E1FLA5/2002/v2n2/E1FLA5_2002_v2n2_122.pdf
https://arxiv.org/ftp/arxiv/papers/1506/1506.06832.pdf

26-04 12:00-17:00
Gelezen en een korte samenvatting gemaakt zodat ik een overzicht heb en kan zien wat ik per paper nodig heb voor mijn eigen onderzoek:
https://www-sciencedirect-com.proxy.library.uu.nl/science/article/pii/S0167639302000833
file:///C:/Users/Rens%20Kersbergen/Downloads/05_479-488_00827.pdf
https://www.researchgate.net/profile/Sefik_Eskimez/publication/327811768_Unsupervised_Learning_Approach_to_Feature_Analysis_for_Automatic_Speech_Emotion_Recognition/links/5c4f35fe458515a4c746c877/Unsupervised-Learning-Approach-to-Feature-Analysis-for-Automatic-Speech-Emotion-Recognition.pdf
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.385.437&rep=rep1&type=pdf
https://www.montana.edu/attmemlab/documents/hupkaetal1999.pdf

28-04
10:00-12:00
Set up Git https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup
$ git config --list
diff.astextplain.textconv=astextplain

filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
credential.helper=manager
user.name=Rens Kersbergen
user.email=renskersbergen@hotmail.nl
alias.co=checkout
alias.br=branch
alias.st=status

12:00-13:00
Geprobeerd voorbeeld script te runnen. De meeste voorbeeldscripts vereisen bestanden van het Linguistic Data Consortium (zoals wsj).

30-04
10:00-15:00
Onderzoeksvraag en deelvragen aangepast

05-05
10:00-15:00
Geprobeerd voorbeeldscript chime1 chime2 en chime3 te runnen. De bestanden nodig voor chime1 zijn te groot en bestanden voor chime2 en chime3 vereisen toegang tot het linguistic data consortium.

07-05
10:00-17:00
Onderzoek gedaan naar Mel Frequency Cepstral Coefficient
https://arxiv.org/ftp/arxiv/papers/1506/1506.06832.pdf

12-05
10:00-17:00
Onderzoek gedaan naar chroma features. Beslissing om Kaldi te gebruiken.

14-05
10:00-17:00
Geprobeerd voorbeeldscript yesno in Kaldi te runnen. De functie wget werkt niet met windows. Presentatie voorbereid. Pytorch kaldi installatie via
https://github.com/mravanelli/pytorch-kaldi/blob/master/README.md#how-to-install

19-05
10:00-17:00
MS Teams meeting met presentatie. Gezocht naar alternatief voor wget.

21-05
10:00-17:00
Werking van SER python script onderzocht. Werking van alle kaldi folders opgezocht. Pytorch Kaldi installatie via https://github.com/mravanelli/pytorch-kaldi/blob/master/README.md#how-to-install

26-05
10:00-17:00
Onderzoek naar missende onderdelen in SER-python script en mogelijke potentiele verbeteringen. Enkele parameters zijn aangepast en de beste combinatie van emotie categorieën voor de meest optimale score gevonden.

28-05
10:00-17:00
Onderzoek naar Pytorch en Kaldi in het project Pytorch-Kaldi en onderzoek naar verschillende emotieherkenningstechnieken in Kaldi.

02-06
10:00-17:00
Het formuleren van werking en relevantie neurale netwerken.

04-06
10:00-17:00
Nieuw plan gemaakt voor mijn onderzoek. Zoeken naar geschikte onderzoeken om eigen python experiment mee te vergelijken. Meerdere onderzoeken gevonden die verschillende SER-technieken toepassen en twee papers van Sarma gevonden die architectuur en datas heeft getest op SER in Kaldi.

06-06
11:00-18:00
Sarma 2018 en 2019 gelezen en samengevat.

09-06
10:00-17:00
Data verzamelen uit de SER Python script en deze vergelijken met het onderzoek van Sarma (2018)

11-06
10:00-17:00
Vergelijken en beschrijven van RAVDESS en IEMOCAP databases.

13-06
10:00-17:00

Samenvatten van onderzoeken die verschillende SER technieken toepassen. Ik verwacht niet dat ik alle technieken zal beschrijven in mijn paper, maar ik zet de meest relevante technieken alvast in de tekst van mijn eerste versie.

14-06
10:00-21:00
Verder met samenvatten van onderzoeken die verschillende SER technieken toepassen. Conclusie maken uit mijn bevindingen en de eerste versie klaar maken voor inleveren (lay out, APA referenties, etc.).

15-06
6:00-7:00
Eerste versie doorgelezen en ingeleverd.

17-06
10:00-15:00
Feedback ontvangen en gezocht naar oplossingen voor de problemen die zijn belicht door mijn begeleider. Sommige problemen zijn makkelijk op te lossen terwijl andere problemen een nieuwe strategie vereisten.

20-06
10:00-17:00
Perturbatie toegepast op alle trainingsdata.

21-06
10:00-15:00
Python script aangepast voor nieuwe trainingsdata
.
22-06
13:00-17:00
Nieuwe experimenten uitgevoerd met nieuwe trainingsdata en resultaten genoteerd. Conclusie gemaakt op deze nieuwe resultaten.

23-06
10:00-17:00
Feedback van mijn begeleider verwerkt en nieuwe versie van scriptie verzonden naar een paar vrienden en familie. Ik ben vooral benieuwd of mijn verwoording in de introductie duidelijk genoeg is voor iemand die het nog nooit gelezen heeft.

25-06
12:00-13:00
Lay-out aangepast conform APA

26-06
11:00-16:00
Conferentieposter gemaakt.