ARL-TR-9137 ● JAN 2021

DEVCOM ARMY RESEARCH LABORATORY

Adapting a Model of Emotional State Recognition to Detect Stress in a High-Noise Environment

by Angelique Scharine and Kristin E Schaefer

Approved for public release; distribution is unlimited.

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9137 ● JAN 2021



Adapting a Model of Emotional State Recognition to Detect Stress in a High-Noise Environment

Angelique Scharine
Human Research and Engineering Directorate,
DEVCOM Army Research Laboratory

Kristin E Schaefer
Vehicle Technology Directorate,
DEVCOM Army Research Laboratory

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
January 2021	Technical Report	September 2019–November 2020

4. TITLE AND SUBTITLE
Adapting a Model of Emotional State Recognition to Detect Stress in a High-Noise Environment

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Angelique Scharine and Kristin E Schaefer

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLH-FD
Aberdeen Proving Ground, MD 21005

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-9137

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
ORCID IDs: Angelique A Scharine, 0000-0001-8230-1824; Kristin E Schaefer, 0000-0002-1342-3446

14. ABSTRACT
A convolutional recurrent neural network model was implemented to detect the state of crewmembers during human–autonomy teaming as a means to quantify changes in affect-based team trust and to further understand team dynamics. The model, based on previously developed speech emotion-recognition models, was developed to classify recorded audio as nonspeech, neutral, or engaged. The model was trained and validated using audio recordings from the Wingman field exercise for manned–unmanned lethality operations. Despite the presence of significant amounts of engine noise, the model performed accurately, classifying more than 99% of the test data correctly. These findings are discussed in terms of future steps required to implement the model to identify when Soldiers are engaged and adaptive autonomous assistance is required. Being able to detect when Soldiers are engaged in operational tasks allows system designs in which autonomous agents can prioritize tasks and information and take autonomous control of others. This, in turn, may increase trust and cohesion within human–autonomy teams.

15. SUBJECT TERMS
speech emotion recognition, convolutional recurrent neural network, human–autonomy teaming, cognitive load, adaptive autonomous assistance

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Angelique Scharine
Unclassified	Unclassified	Unclassified	UU	56	19b. TELEPHONE NUMBER (Include area code) 410-278-5957

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures v

List of Tables v

Summary vi

Acknowledgments vii

1. Introduction 1

1.1 Team Dynamics 1

1.2 Communication and Team Trust 3

1.3 Current Work 3

2. Background: The Detection of Emotional State from Speech 4

2.1 Application of Machine Learning Methods to Speech Emotion Recognition 4

2.2 A Model Based on 3-D Features, CRNN, LSTM, and Attentional Weighting 7

2.3 Algorithmic Methods to Enable Technology Adaptation to Human Crewmembers 8

3. Methods 9

3.1 Field Exercise: Wingman (October 2018) 9

3.2 Speech Stimuli 9

3.3 File Structure 10

3.4 Model Structure 12

4. Results 14

5. Discussion 16

5.1 Speech Emotion Recognition for Teams Composed of Human and Autonomous Members: Trust and Cohesion 16

5.2 Path Forward 17

6. Conclusions 18

7. References 19

Appendix A. Code Sample (zscore_WMtrin.py) 27

Appendix B. Code Sample (ExtractMel_WMtrin.py) 34

Appendix C. Code Sample (SpeechEmotClassifierWMtrin.py) 43

List of Symbols, Abbreviations, and Acronyms 46

Distribution List 47

List of Figures

Fig. 1 Sample folder of wave files corresponding to the data shown in Table 1 11

Fig. 2 3-D attention CRNN model for SER 14

Fig. 3 Plot of accuracy obtained for each of 1000 training epochs. The orange line is the performance on the validation set. Performance of the trained model on the test set was also 1.0000. 15

List of Tables

Table 1 Wingman speech data 10

Table 2 Sample text file containing data labels 11

Table 3 Summary of compiled model 12

Table 4 Model history for the first 10 and last 5 epochs (live data only) 15

Summary

This research was conducted to investigate the feasibility of using acoustic information from speech to detect when adaptive autonomous assistance is needed to calibrate human–autonomy team effectiveness. Machine learning techniques, in particular convolutional recurrent neural network models, were used for this study and evaluated for their ability to realize the real-time benefits of this approach. While this model is one of several possible ways to enable autonomous agents to detect and adapt to a changing operational environment, it has potential to detect nonverbal information in verbal communication. This approach allows humans to work more cohesively with their autonomous partners and to trust that these autonomous agents can determine mission context from speech as a human teammate would, enabling them to prioritize information and tasks appropriately.

A use case from the Wingman Joint Capabilities Technology Demonstration for manned–unmanned gunnery operations was used as the training set to develop a convolutional recurrent neural network model to identify and classify speech as either "nonspeech", "neutral", or "engaged". The key outcome reported here is that the model correctly identified speech samples at a rate greater than 99% despite being operated in a noisy environment and having limited training data. The path forward will be to develop a working version of this classifier for use in real-time speech classification.

Acknowledgments

This work supports the US Army Wingman Joint Capability Technology Demonstration (JCTD) and the US Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory's Human–Autonomy Teaming Essential Research Program. The authors acknowledge Mr Ralph Brewer (Vehicle Technology Directorate), Dr Anthony L Baker (Oak Ridge Associated Universities), and Ms Debra Patton (currently with the DEVCOM Data & Analysis Center) for their assistance with data collection during the 2018 Wingman JCTD field exercise. We also acknowledge DEVCOM Ground Vehicles System Center and the Wingman JCTD for allowing us to use audio recordings for development of this model. The authors also thank Drs Joe Rexwinkle and Nicholas R Waytowich and Mr Jonroy D Canady for their helpful comments and suggestions on previous drafts of this report.

1. Introduction

As part of the US Army modernization priorities, the Army proposes to integrate autonomous systems, such as the autonomy-enabled assets that comprise the Next Generation Combat Vehicle (NGCV) concept, with the Warfighter to augment and enhance Soldier and team performance in increasingly complex environments (South 2019). Ideally, autonomous systems will provide several advantages, such as reducing threats to Soldiers, providing additional distributed sensor capabilities to increase situation awareness, and enabling faster, more-informed decision-making. As such, a goal of the US Army Futures Command and the US Army Combat Capabilities Development Command Army Research Laboratory's Human–Autonomy Teaming (HAT) Essential Research Program is to address the complexities of teaming humans and autonomous systems within an already complex tactical environment in order to create synergistic teams that function effectively and adapt to the dynamic nature of combat.

The Wingman Joint Capabilities Technology Demonstration (JCTD) and the Robotic Combat Vehicle are two programs supporting NGCV development of armored vehicles, such as the Mission Enabler Technologies–Demonstrators and the robotic Wingman vehicles, for military use. Unlike legacy vehicles, these are designed to be operated in teams that include crews of Soldiers in the control vehicle teamed with remote weaponized robotic vehicles. While this will increase the lethality of such teams, it will also increase the responsibilities of its crewmembers. Thus, it is essential to understand the intricacies of how humans and autonomy interact to effectively develop team cohesion and calibrate trust.

1.1 Team Dynamics

As these types of robotic systems continue to move from teleoperated tools to intelligent team members (Phillips et al. 2011), it is critical to understand the new team dynamics that emerge. For example, a major goal of integrating intelligent agents is to reduce or offset human "overload".[*] However, a major challenge is balancing optimal sharing or trading of control between the human(s) and autonomous tools or entities (Miller & Parasuraman 2007) with the function allocation or design decisions determining how to distribute tasks between humans and autonomous agents to achieve the mission goal (Deng et al. 2020). This is further complicated by the fact that changes in control allocation should be

[*]"Overload" here is loosely defined. For the purposes of this report, I prefer the term "engaged", meaning "busy". The intent, therefore, is to allow autonomy to detect when operators are busy and react appropriately.

dynamic, meaning the level of control should adjust given the current capabilities of the team members (both human and robot) at given times throughout a mission.

One advantage of changing levels of autonomy is that it allows the system to optimize the operator's levels of stress and trust. Obviously, cognitive overload leads to errors and fatigue. However, too little stimulation can lead to complacency and missed information. By modulating overload and underload, it may be possible to extend the operators' ability to remain vigilant (Heard and Adams 2019). Notwithstanding, changing the level of autonomy also has costs. Switching from automatic to manual control regularly may result in the operator losing track of which control scheme is active and relying on the automation inappropriately. Conversely, the operator may fail to use automation when it is needed and available. However, familiarity with the manual operation of a system also allows operators to better supervise and monitor the autonomous performance (Di Nocera et al. 2005; Kircher et al. 2014; Chavaillaz and Sauer 2017; Deng et al. 2020).

Dynamic function allocation allows a system to adapt to the cognitive load of its human team members, as well as the complexity of the mission and the capabilities of the technology. There are two types of dynamic function allocation: adaptable and adaptive. *Adaptable function allocation* means the human is in control of the level of autonomy, while *adaptive function allocation* refers to autonomy-controlled function allocation, in which the system uses algorithms to detect when greater assistance is needed (Parasuraman et al. 1996, 2009; Inagaki 2001; Sheridan 2011; Calhoun et al. 2011; Sauer et al. 2012; Chavaillaz and Sauer 2017). To optimally allocate functions, it is first necessary to identify areas where the human team members need assistance. These can be categorized as information overload, distributed decision-making, error prevention, and highlighting critical information (Cummings et al. 2010):

1) *Information Overload*. Use autonomy-enabled information acquisition, filtering, and analysis (Parasuraman et al. 2000) to classify and prioritize incoming information.

2) *Distributed Decision-Making.* The use of automated information acquisition and interpretation, such as automated analysis of sensor feeds and making model-based predictions, can be used to enable, and even automate, more-effective decision-making (Parasuraman et al. 2000). Autonomy-enabled systems can distribute decision-making among team members and ensure that the relevant information and criteria are available to the appropriate members of the team (Nevatia et al. 2008).

3) *Error Prevention.* Autonomy-enabled systems can use methods such as querying the decision-maker to account for human errors due to the use of

heuristics and biases. Sensors and sensor analysis can improve the available information and enhance team operations.

4) *Highlight Critical Information.* Autonomous entities can augment displays in a number of ways that draw operator attention to key information required to successfully perform their role (e.g., image analysis of potential threats, aided target recognition [Nevatia et al. 2008]). Further, autonomy-enabled systems can be equipped with information about the priorities of different events and communicate information or perform tasks according to those priorities.

1.2 Communication and Team Trust

Within teams, communication is critical to maintaining a common understanding (Clark and Brennan 1991), as it is the vehicle through which team members share information, align toward goals, and resolve disagreements (Salas et al. 2005). However, when teams fail to communicate effectively, they can fail to work together effectively. Previous research has shown that identification of communication-based patterns reveal how a team works together to complete tasks, distribute information, and coordinate goal-directed behaviors (Sacks et al. 1974; Kiekel et al. 2001; Tiferes et al. 2016). Team communication patterns offer useful data that, when analyzed, can provide key insights into team coordination, cohesion, and trust (Baker et al. 2020).

1.3 Current Work

This work extends beyond prior research that has begun to identify communication-based metrics associated with team trust and cohesion (see Baker et al. 2020; Baker et al. in press; Krausman et al. submitted) to identify how crew acoustic information (rather than content) can be used to identify emotion-based features that can signal when the autonomous agents should adaptively assist by automating and prioritizing certain tasks. Here, we describe the initial development and testing of a convolutional recurrent neural network (CRNN) to identify and classify emotional state from speech.

The relationship between emotional state, as measured from speech, and trust or cohesion is likely complex. A cohesive, trusting team is likely to display more positive emotions relative to a less cohesive team (Thompson 2016). However, task engagement, as measured through speech, has a more complicated relationship with team dynamics because it is more likely to reflect the specific needs of the situation and the individual's role in that situation than the team's status. Despite this, it has great promise as a tool to improve the cohesiveness and trust within a team when

used as a method of determining when to intervene autonomously. Essentially, the ability to detect engagement enables autonomous agents to deliberately take over less-complex tasks or assist in cognitive tasks by clarifying and providing information about task options. Further, as a means for after action review, the detection of high-engagement periods facilitates analysis to better understand the sources of cognitive overload.

2. Background: The Detection of Emotional State from Speech

The emotional intelligence of a team predicts the efficacy of its performance (Elfenbein 2013). Specifically, team members with high levels of emotional intelligence are able to modulate their behaviors by perceiving and adapting to the emotional state of other team members (Waldinger et al. 2004). We learn to implicitly perceive the emotions of others by reading subtle cues, including facial expression and changes to the voice (Neubauer et al. 2017). This allows us to adapt our own communications and actions. If a team member is overloaded, another team member can assist with tasks, thus dividing the workload more evenly. If a team member is angry, the other team member may act in ways to appease the other; this ability to adapt to others is a useful human skill. Good teamwork often consists of knowing when to pass along information and knowing when this information will not be helpful due to the constraints of the recipient's state (Lingard et al. 2004).

Operational platforms are being designed today that can provide adaptive autonomous assistance (AAA), taking over elements of tasks to reduce the operator's operational load. Although autonomous agents may be very reliable in the tasks they are programmed to do, they are not currently programmed to "read the room"; that is, to adapt their actions based on those of other team members. Ideally, autonomous agents would alter their behavior based on the needs of other team members. During times of stress, human team members may not have the capacity to identify and request assistance effectively. Tunnel vision is an adaptive phenomenon during times of cognitive overload; humans focus on a limited set of tasks to the exclusion of others when overloaded (Keller et al. 2017). Therefore, to facilitate cohesive human–robot teamwork, it is necessary to identify when assistance through automation and cognitive streamlining is needed.

2.1 Application of Machine Learning Methods to Speech Emotion Recognition

Machine learning (ML), deep neural network (DNN), and artificial intelligence techniques have been used successfully to identify the emotional state of human

talkers using the acoustical features of their voices (Casale et al. 2008; Stuhlsatz et al. 2011; Koolagudi and Rao 2012). ML uses a dataset to derive the underlying probabilities about the relationship of the data's features to a given outcome (Bishop 2006). A predictive model is developed by solving mathematically for parameters that predict the data most accurately with the least error.

In the simplest version, an ML classifier is trained using a database of labeled speech items. The recorded speech items are divided into overlapping segments (windows), and these are analyzed to compute several features, thus creating a feature vector for each signal (Ververidis and Kotropoulos 2006). These features usually include information about the spectral content, the log-energy content, the pitch period, and statistical information about their averages, minima, and maxima. Derivatives of these features provide information about feature changes over time (Schuller et al. 2003; El Ayadi et al. 2011). Thus, the task of the ML algorithm is to estimate the model parameters from the data, determining the weight of each feature with respect to the probability of an outcome, and by optimizing these weights, the model can classify the input into categories.

Speech emotion recognition (SER) researchers have extensively compared features to determine which are most predictive and how best to optimize the parameters (for a review, see Basharirad and Moradhaseli 2017). Typically, longer window segments are preferable for emotion detection than those used for speech recognition (Luengo et al. 2010; Ramakrishnan 2012; Koolagudi and Rao 2012; Albornoz et al. 2014) as the features that signal emotional state are more likely to be found in the changes in the voice over time (Luengo et al. 2005; Bitouk et al. 2010). Although initial models used hidden Markov models or support vector machines, these make predictions based on each feature independently and do not account for the interactions of features over time. While it is possible to obtain somewhat accurate estimates (>60%) from utterance-level features, it seems that most of the information about emotional state is found in the higher-order features (Casale et al. 2008; Bitouk et al. 2010; Chen et al. 2018; Zhang et al. 2018). For this reason, researchers began to investigate approaches that use multiple layers of parameters, such as hierarchical structures, decision trees, and neural network (NN) models.

There are several databases of labeled speech recordings (Fiscus et al. 1993; Hansen et al. 1997; Ververidis and Kotropoulos 2003, 2006; Burkhardt et al. 2005; Swain et al. 2018), but the speech datasets used for ML and NN modeling have significant limitations. None of these are very large, at least compared with what is normally used for ML, and most rely on actors to produce the different emotional elements (Vogt and André 2005; Cao et al. 2015). Further, the assumption is that most speech is neutral, which means that the detection of emotions is essentially the detection

of outliers. Some emotions are difficult to distinguish from others; however, these models are able to correctly classify speech signals at rates well above chance (Casale et al. 2008; Fayek et al. 2017). They are most successful in discriminating between positive and negative emotions (Casale et al. 2008). They have been used for several applications, including use in psychiatric diagnoses, customer service systems, and voicemail systems (Yacoub et al. 2003; Cannizzaro et al. 2004; Vidrascu and Devillers 2005; Vogt et al. 2008; Vergyri et al. 2015; Banerjee et al. 2017).

One inherent problem with the earlier ML methods is that they do not take into account time; that is, the classification of one window of speech does not depend or interact in any way with the speech that has occurred before or will occur after that segment. This leads to issues because the features associated with emotional state are context-dependent, and the baseline may vary from one context to another and from one speaker to another (just as with automatic speech recognition). For this reason, the development of deep-learning techniques has been quite useful for emotional state recognition from speech (Stuhlsatz et al. 2011; Kim et al. 2013; Albornoz et al. 2014; Schmidhuber 2015). For example, DNNs incorporate multiple layers of abstraction, allowing the network to use both lower-level and higher-level features. Information is transformed from one layer to the next using a feed-forward architecture.

Taking these elements a step further, convolutional neural networks (CNNs) transform data using a convolutional kernel followed by a pooling layer. The convolutional kernel acts as a filter, extracting different levels of features. The pooling layer reduces the number of similar or correlated features by subsampling. Recurrent neural networks (RNNs) incorporate connections to the layer's previous state, thus allowing the model parameters to incorporate successive inputs, making them ideal for speech (Badshah et al. 2017; Lim et al. 2017; Chen et al. 2018). One form of RNN is that of long short-term memory (LSTM), which incorporates a forgetting parameter, sometimes called momentum, which determines the weight of the previous input on the current parameter. In this way, the parameters are more dependent on recent speech since the influence of earlier speech is reduced over time.

In the implementation described here, the detection of emotional state will be achieved using a CRNN. As noted previously, this model architecture incorporates successive temporal features allowing it to make predictions based in part on context. The convolutional kernels serve to reduce the processing requirements of running the model by allowing parameters to be shared across layers, thus reducing the number of parameters requiring estimation. In particular, the model proposed by Chen et al. (2018) was used as the basis of this project.

2.2 A Model Based on 3-D Features, CRNN, LSTM, and Attentional Weighting

Chen et al.'s (2018) model is an extension of previous CRNN models first developed by Huang and Narayan (2016). His model introduces the use of an attention mechanism that weights the portions of speech that contain emotion-relevant information more heavily (Huang and Narayanan 2016, 2017; Neumann and Vu 2017; Mirsamadi et al. 2017). This attention component, in combination with CRNN, accounts for individual speaker differences.

Chen's team describes this model as a 3-D model. It uses as input the static log-Mels and their derivatives, delta (magnitude of change) and delta-delta (rate of change). This input provides both spectral and temporal information about the speech signal, effectively reducing the effect of emotion-irrelevant factors, such as the style of speaking, speech contents, and the environment. This part of the model uses code developed by James Lyons that computes the filterbank energies from the signal (Lyons 2017).

These log-Mel filterbank energies contain information about the power spectrum of speech. The signal is passed through a pre-emphasis filter, gets sliced into (overlapping) frames, and is filtered with a windowing function. Each frame is Fourier transformed and the power spectrum is computed. From these the filterbanks are computed. Essentially, the log-Mel filterbank process is designed to capture elements of the speech signal while accounting for human perception. The derivatives capture the change in this content, giving relative information about the changes in speech over time.

This gives a 3-D feature representation, $\in R^{t \times f \times c}$, as input to the CNN model, where t denotes the frame length, f denotes the Mel-filter bank number, and c denotes the number of feature channels. This input is convolved with several 3-D convolution layers, one 3-D max-pooling layer, one linear layer, and one LSTM layer. Chen's team varied the number of convolutional layers and found the best performance for five or six layers, depending on the speech database used (Chen et al. 2018). Max-pooling after the first convolutional layer reduces the number of parameters by a factor of four. The output from the linear layer is then fed into a bidirectional LSTM (BLSTM) layer for temporal summation. This allows the model to incorporate time-series information as context for the current utterance. The LSTM output is then passed through the attention layer, where the features are scored based on their importance using a soft-max function and then a weighted sum.

2.3 Algorithmic Methods to Enable Technology Adaptation to Human Crewmembers

The objective of this work was to develop an algorithm capable of real-time detection of the emotional state of crewmembers that signals when AAA is required, supporting accurate changes of control allocation dependent on crewmember state. For example, identification of inappropriate crewmember stress can be used as a signal for when autonomous agents could take over some routine functions and prioritize the information it provides the user. It has been shown that it is possible to detect the emotional state from the speaker's voice; however, the military context differs from that of the speech databases that have been used in past studies in that it is a considerably noisier and likely more stressful environment. Further, the vocabulary is more constrained, as Warfighters are trained to use specific terms and to repeat and acknowledge commands.

Previous research has shown that it is easier to train a model to discriminate between positive and negative states than between finer distinctions. Casale et al. (2008) found that their model accurately detected (95% accuracy) a negative category consisting of anger, boredom, disgust, fear, and sadness. Several have reported difficulty distinguishing between "happy" trials and "neutral" or "surprise" (Chen et al. 2018). For this application, there is little advantage in correctly classifying all of the emotions. It is more important to detect when the speaker is cognitively loaded—that is, busy. Therefore, the model was trained to distinguish between nonspeech and two speaking states: neutral and engaged.

The first step was to program a CRNN model similar to that of Chen et al. (2018) and verify that it can successfully discriminate between emotions from a known speech dataset. For this purpose, we used the Interactive Emotional Dyadic Motion Capture (IEMOCAP) database (Busso et al. 2008). Once a working model was coded, the items in the IEMOCAP database were recoded into positive or negative emotions. Items labeled as "fear" or "anger" were recoded to "negative" and those with any other emotion were labeled as "neutral". As expected, the model was more accurate when making a binary decision.

One major difference between this application of the model and previous ones is the presence of noise, specifically from the combat vehicle in which the speech occurs. With relatively few exceptions, previous models have been implemented on databases containing clean speech recordings. At most, the level and type of noise is more like that of office noise than that of a military vehicle (Vidrascu and Devillers 2005; Eyben et al. 2016). Therefore, the discrimination is not just between neutral or engaged speech; the model must detect when speech is present as well. Perceptually, the neutral and engaged speech categories are more similar than the

"nonspeech" condition. Therefore, the model proposed here is trained to discriminate between three conditions, "nonspeech", "neutral", and "engaged".

3. Methods

3.1 Field Exercise: Wingman (October 2018)

Speech recordings were made during the October 2018 Wingman JCTD event held at Fort Benning, Georgia. The Wingman program developed prototype manned and unmanned vehicle platforms (high-mobility multipurpose wheeled vehicles [HMMWVs][†]) for use in joint manned–unmanned team gunnery tasks. The command and control (C2) vehicle carried the crew, while the robotic combat vehicle was unmanned and weaponized. During the JCTD event, two five-man crews (one Marine crew and one Army crew) conducted the live-fire gunnery evaluation exercises. (Schaefer et al. 2018; Brewer et al. 2019).

The field exercises followed the standard engagement process, known as DIDEA (detect, identify, decide, engage, and assess). This required the teams to acquire targets, identify them as potential threats, make a decision to engage or not to engage a target, engage the target(s), and then assess the effects of each firing action. The crews were trained on the command terminology. The training process included a dry run (no ammunition) and multiple blank-fire runs (blanks). After training, the Soldiers and Marines (all male) took part in multiple stationary live-fire engagement runs. These runs were like the training runs but with two differences: live ammunition was used and, to comply with safety requirements, the robotic combat vehicle was chocked to prevent vehicle movement while the weapon was loaded.

3.2 Speech Stimuli

Audio recordings were made during all of the training and live runs; however, due to technical issues, some of these recordings were not of sufficient quality for use. Recordings from only one dry run and one live-fire run for both the Soldiers and Marine were of sufficient quality for analysis. These recordings were previously used and are described in an earlier analysis of communication patterns (Baker et al. 2020).

[†]The current vehicles selected for this program are a Stryker C2 vehicle for crew operations and an M113 robotic combat vehicle. The HMMWVs used during this exercise were from an earlier prototype.

The original recordings were made in stereo at a 44.1-kHz sampling rate with 16-bit quantization. Each track was down sampled to 16 kHz and split into two monaural tracks. Segments from both tracks were used for training and testing. Each recording was subdivided into 2-min segments. Each segment was labeled at millisecond precision to mark when speech started and ended. Each recorded segment was manually labeled as nonspeech, neutral or engaged. Because speech is not continuous, nonspeech segments with pauses in speech shorter than 2 s were not considered breaks in speech. Some of the spoken segments were very short, with the minimum segment length for spoken segments being 300 ms. Nonspeech segments were segments with no speech for longer than 2 s. In some cases, these contained throat clearing or coughing. An engaged segment meant that the team was actively engaged in one of the target runs. Neutral segments contained casual conversation and sometimes discussion of issues experienced during the test. Some descriptive statistics for the wave-files are shown in Table 1.

Table 1 Wingman speech data

Folder name	Test 2	Test 3	Test 5	Test 6	All four tests
Type	Live run	Live run	Dry run	Dry run	…
Participants	Marines	Army	Marines	Army	…
Duration	31:12.000	29:29.000	26:03.165	17:11.677	1:43:55.842
Count nonspeech segments	280	220	172	118	780
Count neutral segments	148	146	102	64	460
Count engaged segments	176	92	108	92	468
Count all segments	604	458	382	274	1708
Mean duration nonspeech	00:07.617	00:08.766	00:11.148	00:09.816	00:09.038
Mean duration neutral	00:04.192	00:04.931	00:05.261	00:04.739	00:04.740
Mean duration engaged	00:03.646	00:05.079	00:03.881	00:03.969	00:04.045

Note: The test numbers refer to the numbers used in the overall Wingman program, a subset of which were used in this analysis. They are intended to be nominal descriptions, as sequence is unimportant.

3.3 File Structure

The wave-file for each field exercise was divided into 2-min segments that were further divided into subsegments, each labeled with a single state category. A text file was created for each 2-min segment that contained the wave-file names corresponding to the subsegments and their labels (Table 2). The data folder for each test contained two subfolders, one for the text files (EmoEval) and one for the wave-files (wav). Within the wave-file folder were subfolders that corresponded to

each text file. Each of those subfolders contained the wave-files described in that text file (Fig. 1). This larger dataset was divided into three subsets, 70% of which were used for training, 15% for cross-validation, and 15% for testing. The items within these sets were randomly chosen from the overall set to contain equivalent proportions of nonspeech (nsp), neutral (neu), and engaged (eng) items.

Table 2 Sample text file containing data labels

Start time–end time	Wavfile name	Label
[0:00.000 – 0:01.221]	part00_00	neu
[0:01.222 – 0:18.255]	part00_01	nsp
[0:18.256 – 0:45.765]	part00_02	neu
[0:45.766 – 0:50.238]	part00_03	nsp
[0:50.239 – 1:05.050]	part00_04	neu
[1:05.051 – 1:09.999]	part00_05	nsp
[1:10.000 – 1:30.216]	part00_06	neu
[1:30.217 – 1:34.714]	part00_07	nsp
[1:34.715 – 1:36.000]	part00_08	neu
[1:36.001 – 1:44.954]	part00_09	nsp
[1:44.955 – 1:48.908]	part00_10	neu
[1:48.909 – 1.54.246]	part00_11	eng
[1:54.247 – 2:00.000]	part00_12	nsp

Note: File folder name/tests/test2/EmoEval/part00

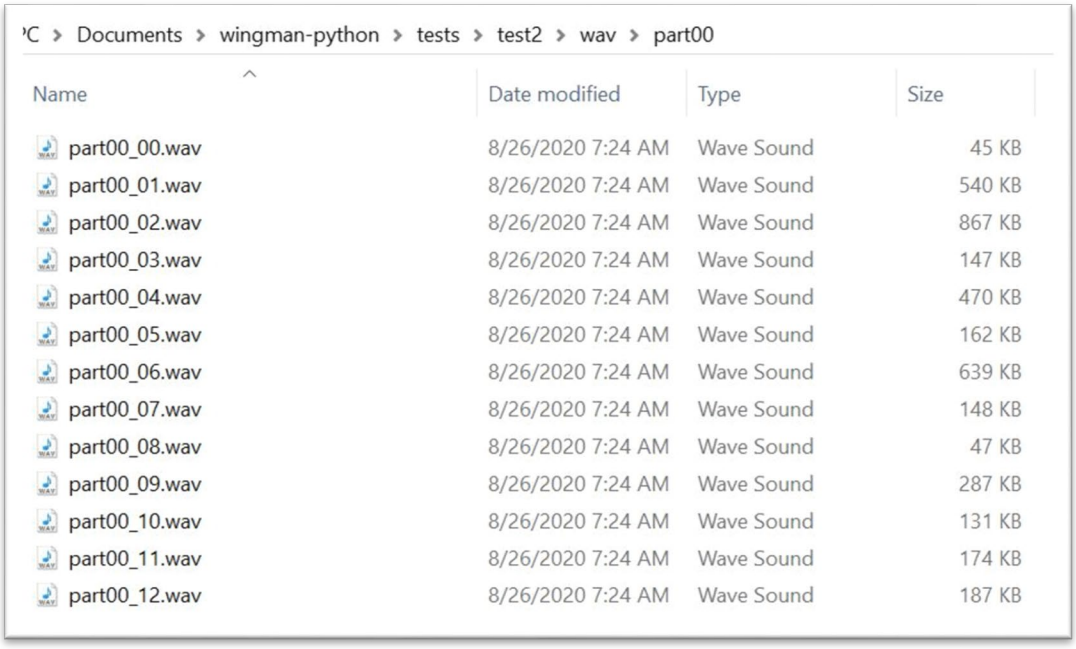


Fig. 1 Sample folder of wave files corresponding to the data shown in Table 1

3.4 Model Structure

The model is a standard CRNN model similar to the SER models implemented recently (Huang and Narayanan 2016; Zhang et al. 2017) that use log-Mels. However, unlike these models, which are trained on individual speech patterns, this one adopts the use of the derivatives delta and delta-delta (Chen et al. 2018). This allows the model to incorporate the change of the log-Mel information over time as feature inputs, effectively allowing the model to use statistical information from the individual samples as a variable (Table 3).

Table 3 Summary of compiled model

Block	Layer	Filter	Size	Params	Output	Activation	Options
1	Conv2D	5 × 3	128	5888	(:, 296, 38, 128)	re_lu	
	BatchNorm			512	(:, 296, 38, 128)		
	Leaky_re_lu						Leakiness = 0.1
	MaxPool2D	2 × 2			(:, 148, 19, 128)		
	Dropout						p = 0.2
2	Conv2D	5 × 3	256	491,776	(:, 144, 17, 256)	re_lu	
	BatchNorm			1024	(:, 144, 17, 256)		
	Leaky_re_lu						Leakiness = 0.1
	Dropout						p = 0.2
3	Conv2D	5 × 3	256	983,296	(:, 140, 15, 256)	re_lu	
	BatchNorm			1024	(:, 140, 15, 256)		
	Leaky_re_lu						Leakiness = 0.1
	Dropout						p = 0.2
4	Conv2D	5 × 3	256	983,296	(:, 136, 13, 256)	re_lu	
	BatchNorm			1024	(:, 136, 13, 256)		
	Leaky_re_lu						Leakiness = 0.1
	Dropout						p = 0.2
5	Reshape				(:, 256, 1768)		
	Bidirectional LSTM	…	128	1,942,528	(:, 256, 256)		
	Flatten				(:, 65,536)		
	Dense			4,194,368	(:, 64)	re_lu	…
Classifier	Dense	…	…	195	(:, 3)	Softmax	…

First, each speech signal was normalized using its mean and unit variance (Appendix A). Each signal was then split into short overlapping frames with Hamming windows of 25 ms with a 10-ms shift. The power spectrum was computed for each frame using a discrete Fourier transform. This power spectrum was passed through a Mel-filter bank i to produce p_i. The log-Mel m_i was computed by taking the logarithm of p_i (Eq. 1). The derivative, delta, and the

time-derivative delta-deltas are then computed using Eqs. 2 and 3. Here we use $N = 2$. This gives a 3-D feature representation, $X \in R^{t \times f \times c}$ as input to the CNN model, where t denotes the frame length, f denotes the Mel-filter bank number, and c denotes the number of feature channels. Following Chen et al. (2018), we set f as 40 and c as 3.

$$m_i = \log(p_i) \tag{1}$$

$$m_i^d = \frac{\sum_{n=1}^{N} n(m_{i+n} - m_{i-n})}{2\sum_{n=1}^{N} n^2} \tag{2}$$

$$m_i^{dd} = \frac{\sum_{n=1}^{N} n(m_{i+n}^d - m_{i-n}^d)}{2\sum_{n+1}^{N} n^2} \tag{3}$$

CRNN Model

The "ExtractMel" shown in Appendix B was used to extract the 3-D log-Mels for input to the CRNN model (Appendix C). The CRNN used to extract high-level features for SER includes four CNN layers, one 3-D max-pooling layer, one linear layer, and one LSTM layer (Fig. 2). Each direction contained 128 cells, resulting in a 256-dimensional representation of high-level features. The first convolutional layer had 128 feature maps, and each remaining convolutional layer had 256 feature maps. The filter size of each convolutional layer was 5 × 3, where 5 corresponded to the time axis and 3 to the frequency axis. The pooling size for the max-pooling after the first convolutional layer was 2 × 2. To reduce the number of model parameters, the CNN output was flattened into a linear layer with 1768 outputs before passing it to the LSTM layer. Finally, the output from the LSTM layer was fed into a bidirectional RNN with long short-term memory cells (BLSTMs) for temporal summarization. These utterance-level representations were then passed to a fully connected layer with 64 output units to obtain higher-level representations that help the softmax classifier map the utterances into one of the three different emotion classes (nsp, neu, eng). Batch normalization applied to each layer accelerated training and improved the generalization performance.

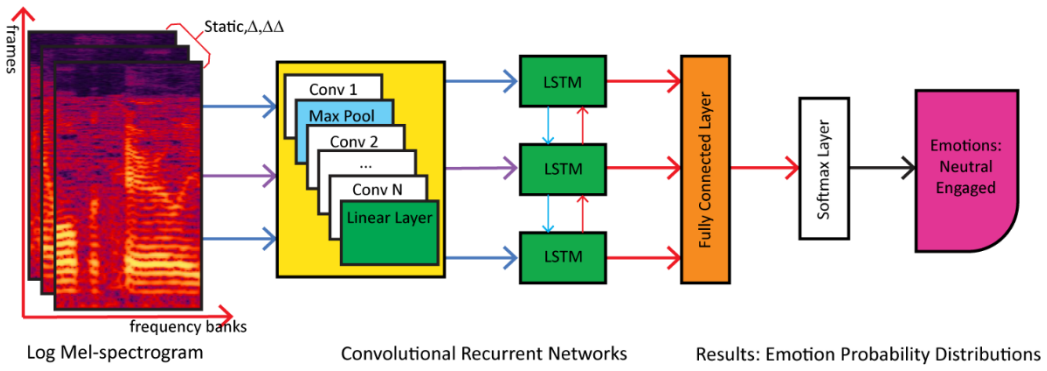


Fig. 2 3-D attention CRNN model for SER

The CRNN architecture was implemented with TensorFlow 2.0 toolkit (Abadi et al. 2016) and Python 3.7 (van Rossum and Drake 2009). The parameter weights of the model were optimized by minimizing the sparse categorical cross-entropy loss function, using the Adam optimizer (Kingma and Ba 2014). The initial learning rate was set to 10^{-4} and the momentum was set to 0.9.[‡] A dropout factor of 0.2 was used to prevent the model from overtraining.

4. Results

Initially, both the live-fire and dry-run recordings were used because this was a relatively small dataset. The model was able to fit parameters to the training set, getting accuracy scores of nearly 100%. However, the accuracy—measured as loss estimates—on the validation and test sets was only roughly 70% for the validation set and 60% for the test set. Although 70% accuracy is generally good performance for a noisy dataset and is well above chance for three classification categories, there was concern that the model was primarily discriminating between the nonspeech and spoken trials.

Because the "engaged" data from the dry-run trials were qualitatively different from the live-fire data (there was no need to evaluate whether the team had successfully hit a target), it may be that the strength of the model's ability to discriminate between engaged and neutral trials was weak because, during the dry run, those trials were very similar. Therefore, the model was subsequently trained, validated, and tested using only the live-run trials. The summary of the first 10 and last 5 epochs are shown in Table 4. The training, validation, and test sets all achieved greater than 99% accuracy (Fig. 3).

[‡]Momentum, in the context of LSTM with temporal context, means that the previous samples are weighted less than the current sample when computing the classification estimate.

Table 4 Model history for the first 10 and last 5 epochs (live data only)

Epoch	Training loss	Training accuracy	Validation loss	Validation accuracy
1	0.7268	0.7142	0.6358	0.7848
2	0.3707	0.8447	2.9755	0.1646
3	0.2518	0.9163	1.6121	0.2447
4	0.1714	0.9385	0.7501	0.7637
5	0.0720	0.9778	0.8656	0.7806
6	0.0603	0.9861	0.0578	0.9831
7	0.0332	0.9924	0.1073	0.9620
8	0.0147	0.9981	0.0338	0.9873
9	0.0050	0.9975	0.0070	0.9958
10	0.0027	0.9987	0.0035	0.9958
996	1.15E-06	1.0000	5.50E-09	1.0000
997	2.50E-07	1.0000	5.00E-09	1.0000
998	7.70E-06	1.0000	4.00E-09	1.0000
999	3.30E-06	1.0000	3.00E-09	1.0000
1000	1.50E-04	1.0000	1.50E-09	1.0000

Note: Performance of the trained model on the test set was also 1.0000.

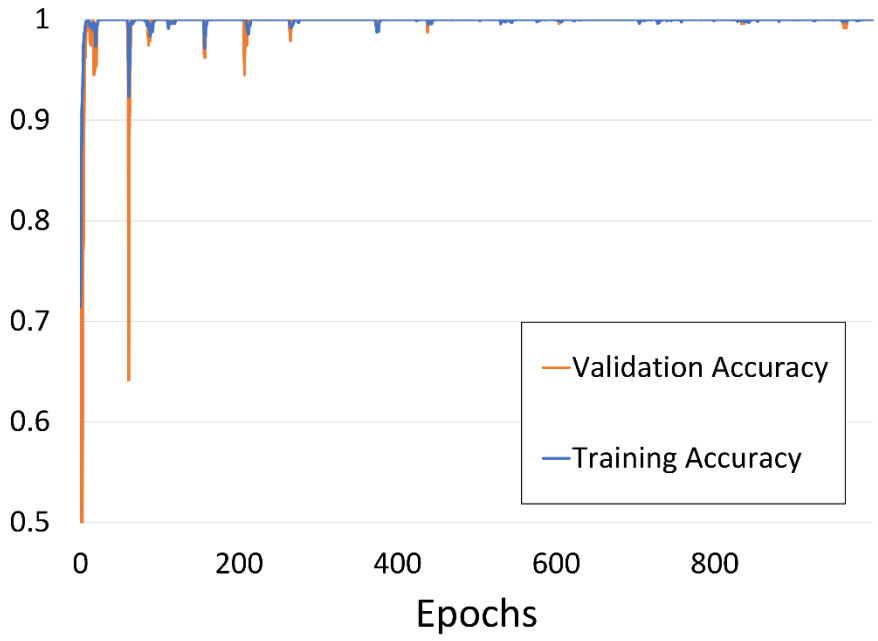


Fig. 3 Plot of accuracy obtained for each of 1000 training epochs. The orange line is the performance on the validation set. Performance of the trained model on the test set was also 1.0000.

5. Discussion

The model clearly distinguished the three categories, making it a promising option for use as an algorithm to detect when Soldiers are engaged. Some factors suggest its generalizability may be overstated and are discussed in the following paragraph. However, the model reaches near perfect performance in just a few epochs, evidence that the three classes are distinct enough to be useful.

Despite the high performance of the model in detecting these classes, the generalizability of this model is limited. Probably the most significant weakness for this model currently is that the size of the dataset is so much smaller than the number of parameters used to fit to it. Another potential weakness is that in using each track of the stereo input as independent samples, there is a chance that the validation and test sets comprise matches to items trained on in the training set. It will be important to the development of this project to obtain more data and test the model's performance, controlling for factors such as potential overtraining. We did take the following steps to, hopefully, reduce overtraining and increase generalizability. The cross-validation process is designed to train the model for optimal generalization to the validation set. The model's near-perfect performance on a new set of data then should indicate generalization—to the extent that the test set was independent of the training set. The use of a drop-out factor (0.2) during training is also a method used to avoid overtraining to the dataset.

However, the generalizability of this model to other speech datasets may also be affected by other factors, such as differences in the vocal characteristics of the operators and differences in tasks. This dataset was trained primarily on male voices, and included only a few samples of female voices. These samples were all from the neutral set. This may introduce bias, in that female samples may be automatically classified as neutral based on the dataset used for training.

Therefore, future work needs to test this model using additional datasets of recorded speech to determine the degree to which the specific vocal characteristics of the operators determine the accuracy of the model. Depending on the level of impact, more training may be needed to create more-robust models that maintain high performance across speakers with varying vocal characteristics.

5.1 Speech Emotion Recognition for Teams Composed of Human and Autonomous Members: Trust and Cohesion

The goal is to identify when AAA is required. This can be implemented in several ways, such as automating navigation, providing decision aids, reallocating responsibilities, and providing visual target-detection aids. A further step is to

implement this model in a test environment and measure team performance as a function of AAA. It is hypothesized that team trust, cohesion, and performance (as assessed by other measures) will improve as a function of the implementation of AAA.

Other research suggests that the ability to adapt one's behavior according to the needs of the team corresponds to greater trust and cohesion within a team and, by extension, to increased team performance (Elfenbein 2013). We have yet to establish that the implementation of AAA in a human–robot team will lead to measurable performance improvements. To do this, we would need to both detect when assistance is needed and implement well-designed technical adaptations. Therefore, we are not currently able to test this hypothesis directly.

Other measures of team trust and cohesion have been identified for use in HAT (Baker et al. in press). If AAA leads to increased team trust and cohesion, scores should increase on these measures as a function of its availability. This should be tested in an environment like that of the Information for Mixed Squads (DEVCOM Army Research Laboratory test facility) study (Chhan et al. 2020; Perelman et al. 2020). This would both validate the use of these metrics in the HAT environment and determine whether AAA, as triggered by SER, will increase team trust and cohesion.

5.2 Path Forward

The following steps need to be taken to develop a prototype system. First, the code needs to be adapted to take a continuous stream of audio data and classify it. Currently, the model was trained, tested, and evaluated on labeled data files that were segmented to separate neutral and engaged content. The next step then is to develop a version of the model that reads in short segments, approximately 4.5 s in length (using overlapping windows), and predicts the state of the speakers. Initially, it does not need to function in real time, but at least continuously.

Looking forward, the objective will be to develop the code such that it can successively read-in data captured with a microphone, capture the Mel-frequency information, and be processed through the model to obtain a classification estimate. Given that it will need to operate in real time, one of the first steps will be to implement a version of the model that reads-in input continuously and outputs a measure of engagedness that can be used to indicate when AAA is needed. The idea would be that if the model detected engagedness for say, 20% of a minute, it would trigger AAA. The specific threshold values will need to be determined through experimentation. This model's performance can be compared with that of the

version using preselected windows to explore the effects using a continuous feed has on the model's accuracy.

Given that the model has been trained on speech recordings from a single training task (two groups of Soldiers), it is possible that the model is overtrained to that specific dataset. It is important to understand how well the model generalizes to other datasets. A similar set of data was collected for the Wingman project in 2019. As with the 2018 study, it incorporated a team of one manned and one unmanned vehicle. By evaluating the accuracy of the model predictions with new data, we can measure the extent to which the model generalizes. Then, the efficacy of this model on larger teams could be assessed by analyzing the Fort Carson mission-enabling technologies demonstrator (MET-D) dataset, where the teams consist of two manned and four unmanned vehicles. A subset of this dataset will be labeled to allow for comparison of the model's performance with the labeled items.

Although the primary intent is to identify when AAA is needed on a real-time basis, it may also prove useful for analysis of crew communications during after-action review. The Fort Carson dataset and others include other measures and information about events during the study. Using the model to identify moments of tension or overload may highlight where processes need reviewing and where problems occur. Further, if used in combination with other measures of team trust and cohesion, it may be possible to measure whether providing AAA affects the teams' cohesiveness in general. These are also objectives for future research.

6. Conclusions

Using speech to predict Soldier load is a novel application of NN modeling. As such, it represents a significant contribution to the Wingman, NGCV, and MET-D programs because it is a method that is not reliant on body-worn sensors. As such, it is less vulnerable to contact and transmission issues. It can work with anyone in the operational environment without outfitting them with specialized sensors. NN-derived models have several potential applications to the combat vehicle environment, from use in speech recognition to target detection to navigation. Thus, it is a tool we may see often in future operations.

The model easily distinguished the three categories of speech (nonspeech, neutral, and engaged), making it a promising option for use as an algorithm to detect when Soldiers are engaged. Although some work needs to be done before it can be implemented in a test vehicle, these results are encouraging.

7. References

Abadi M et al. TensorFlow: a system for large-scale machine learning. 2016. http://arxiv.org/abs/1605.08695.

Albornoz EM, Sánchez-Gutiérrez M, Martinez-Licona F, Rufiner HL, Goddard J. Spoken emotion recognition using deep learning. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics). Springer; 2014. p. 104–111. https://link.springer.com/chapter/10.1007/978-3-319-12568-8_13.

Badshah AM, Ahmad J, Rahim N, Baik W. Speech emotion recognition from spectrograms with deep convolutional neural network. Proceedings of the 2017 International Conference on Platform Technology and Service (PlatCon); 2017 Feb; Busan, South Korea. https://doi.org/10.1109/PlatCon.2017 .7883728.

Baker AL, Fitzhugh SM, Forster DE, Brewer RW, Krausman A, Scharine A, Schaefer KE. Team trust in human-autonomy teams: Analysis of crew communication during manned-unmanned gunnery operations. DEVCOM Army Research Laboratory; 2020. Report No.: ARL-TR-8969.

Baker AL, Fitzhugh SM, Huang L, Forster DE, Scharine A, Neubauer C, Lematta G, Bhatti S, Johnson C, Krausman A, Holder E, Schaefer KE, Cooke NJ. Approaches for assessing communication in human–autonomy teams. Human Intelligent Systems Integration. Special Issue on Human–Autonomy Teaming in Military Contexts. 2021. In press.

Banerjee D, Islam K, Mei G, Xiao L, Zhang G, Xu R, Ji S, Li J. A deep transfer learning approach for improved post-traumatic stress disorder diagnosis. Proceedings of the IEEE International Conference on Data Mining. 2017 Nov; New Orleans, LA. https://doi.org/10.1109/ICDM.2017.10.

Basharirad B, Moradhaseli M. Speech emotion recognition methods: a literature review. AIP Conference Proceedings. 2017;1891(1). https://doi.org/10.1063 /1.5005438.

Bishop CM. Pattern recognition and machine learning. Springer; 2006.

Bitouk D, Verma R, Nenkova A. Class-level spectral features for emotion recognition. Speech Communication. 2010;52(7–8)613–625. https://doi.org /10.1016/j.specom.2010.02.010.

Brewer RW, Cerame E, Pursel ER, Zimmermann A, Schaefer KE. (2019). Manned-unmanned teaming: US Army robotic wingman vehicles. Advances in Intelligent Systems and Computing. 2019;780. https://doi.org/10.1007/978-3-319-94223-0_9.

Burkhardt F, Paeschke A, Rolfes M, Sendlmeier WF, Weiss, B. A database of German emotional speech. Proceedings of the 2005 Annual Conference of the International Speech Communication Association; 2005 Sep; Jeju, South Korea. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.130.8506.

Busso C, Bulut M, Lee CC, Kazemzadeh A, Mower E, Kim S, Chang JN, Lee S, Narayanan SS. IEMOCAP: interactive emotional dyadic motion capture database. Language Resources and Evaluation. 2008;42(4):335–359. https://doi.org/10.1007/s10579-008-9076-6.

Calhoun GL, Ward VBR, Ruff HA. Performance-based adaptive automation for supervisory control. Proceedings of the Human Factors and Ergonomics Society. 2011 Sep. Las Vegas, NV. https://doi.org/10.1177 /1071181311551429.

Cannizzaro M, Harel B, Reilly N, Chappell P, Snyder PJ. Voice acoustical measurement of the severity of major depression. Brain and Cognition.2004;56(1):30–35. https://doi.org/10.1016/j.bandc.2004.05.003.

Cao H, Verma R, Nenkova A. Speaker-sensitive emotion recognition via ranking: studies on acted and spontaneous speech. Computer Speech and Language. 2015;29(1). https://doi.org/10.1016/j.csl.2014.01.003.

Casale S, Russo A, Scebba G, Serrano S. Speech emotion classification using machine learning algorithms. Proceedings of the IEEE International Conference on Semantic Computing; 2008 Aug; Santa Clara, CA. p. 158–165. https://doi.org/10.1109/ICSC.2008.43.

Chavaillaz A, Sauer J. Operator adaptation to changes in system reliability under adaptable automation. Ergonomics. 2017;60(9):1261–1272. https://doi.org /10.1080/00140139.2016.1261187.

Chen M, He X, Yang J, Zhang H. 3-D convolutional recurrent neural networks with attention model for speech emotion recognition. IEEE Signal Processing Letters. 2018;25(10):1440–1444. https://doi.org/10.1109/LSP.2018.2860246.

Chhan D, Scharine A, Perelman BS. Human–autonomy teaming essential research program project 2 : transparent multimodal crew interface designs. Technical note 3: multimodal cueing for transparency in mobility operations. DEVCOM Army Research Laboratory; 2020 May. Report No.: ARL-TN-1019.

Cummings ML, Bruni S, Mitchell PJ. Human supervisory control challenges in network-centric operations. Reviews of Human Factors and Ergonomics. 2010;6(1):34–78. https://doi.org/10.1518 /155723410X12849346788660.

Deng Y, Shirley J, Zhang W, Kim NY, Kaber D. Influence of dynamic automation function allocations on operator situation awareness and workload in unmanned aerial vehicle control. In: Advances in intelligent systems and computing. Springer. 2020 Jan. https://doi.org/10.1007/978-3-030-20040-4_31.

Di Nocera F, Lorenz B, Parasuraman R. Consequences of shifting from one level of automation to another: main effects and their stability. In: Human factors in design, safety, and management. Shaker Publishing; 2005.

El Ayadi M, Kamel MS, Karray F. Survey on speech emotion recognition: features, classification schemes, and databases. Pattern Recognition. 2011;44(3):572–587. https://doi.org/10.1016/j.patcog.2010.09.020.

Elfenbein HA. Team emotional intelligence: what it can mean and how it can affect performance. In: Linking emotional intelligence and performance at work: current research evidence with individuals and groups. Taylor and Francis; 2013. p. 165–184.

Eyben F, Scherer KR, Schuller BW, Sundberg J, Andre E, Busso C, Devillers LY, Epps J, Laukka P, Narayanan SS, Truong KP. The Geneva minimalistic acoustic parameter set (GeMAPS) for voice research and affective computing. IEEE Transactions on Affective Computing. 2016;7(2). https://doi.org/10.1109/TAFFC.2015.2457417.

Fayek HM, Lech M, Cavedon L. Evaluating deep learning architectures for speech emotion recognition. Neural Networks. 2017;92. https://doi.org/10.1016 /j.neunet.2017.02.013.

Fiscus JG, Garofolo JS, Lamel LF, Fischer WM, Pallett DS, Dahlgren NL. DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM. National Institute of Standards and Technology; 1993 Feb. https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir4930.pdf.

Hansen JH, Bou-Ghazale SE, Sarikaya R, Pellom B. Getting started with SUSAS: a speech under simulated and actual stress database. Eurospeech. 1997;97:1743–1746.

Heard J, Adams JA. Multi-dimensional human workload assessment for supervisory human–machine teams. Journal of Cognitive Engineering and Decision Making. 2019;13(3). https://doi.org/10.1177/1555343419847906.

Huang CW, Narayanan SS. Attention assisted discovery of sub-utterance structure in speech emotion recognition. Proceedings of the 2016 Annual Conference of the International Speech Communication Association; 2016 Sep; San Francisco, CA. p. 1387–1391. https://doi.org/10.21437/Interspeech.2016-448.

Huang CW, Narayanan SS. Deep convolutional recurrent neural network with attention mechanism for robust speech emotion recognition. Proceedings of the 2017 IEEE International Conference on Multimedia and Expo; 2017 July; Hong Kong. p. 583–588. https://doi.org/10.1109/ICME.2017.8019296.

Inagaki T. Adaptive automation: sharing and trading of control. Proceedings of the 2001 Transportation and Logistics Conference. https://doi.org/10/1299 /jsmetld.2001.10.79.

Keller MD, Ziriax JM, Barns W, Sheffield B, Brungart D, Thomas T, Jaeger B Yankaskas K. Performance in noise: impact of reduced speech intelligibility on sailor performance in a Navy command and control environment. Hearing Research. 2017;349:55–66. https://doi.org/10.1016/j.heares.2016.10.007.

Kiekel P, Cooke N, Foltz P, Shope S. Automating measurement of team cognition through analysis of communication data. Usability Evaluation and Interface Design. 2001. https://www.researchgate.net/publication/240630991.

Kim Y, Lee H, Provost EM. Deep learning for robust feature generation in audiovisual emotion recognition. Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. 2013 May; Vancouver, Cananda. http://citeseerx.ist/psu.edu/viewdoc/summary?doi =10.1.1.428.5585.

Kingma DP, Ba J. Adam: a method for stochastic optimization. Proceedings of the 3rd International Conference for Learning Representations; 2015; San Diego, CA. Ver. 9; 2017 Jan 30. http://arxiv.org/abs/1412.6980.

Kircher K, Larsson A, Hultgren JA. Tactical driving behavior with different levels of automation. IEEE Transactions on Intelligent Transportation Systems. 2014;15(1). https://doi.org/10.1109/TITS.2013.2277725.

Koolagudi SG, Rao KS. Emotion recognition from speech: a review. International Journal of Speech Technology. 2012;15:99–117. https://doi.org/10.1007 /s10772-011-9125-1.

Lim W, Jang D, Lee T. Speech emotion recognition using convolutional and Recurrent Neural Networks. Proceedings of the 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference; 2016; Jeju, South Korea. https://doi.org/10.1109/APSIPA.2016.7820699.

Lingard L, Espin S, Whyte S, Regehr G, Baker GR, Reznick R, Bohnen J, Orser B, Doran D, Grober E. Communication failures in the operating room: an observational classification of recurrent types and effects. Quality and Safety in Health Care. 2004;13(5):330–334. https://doi.org/10.1136/qshc.2003.008425.

Luengo I, Navas E. Hernaez I. Feature analysis and evaluation for automatic emotion identification in speech. IEEE Transactions on Multimedia. 2010;12(6). https://doi.org/10.1109/TMM.2010.2051872.

Luengo I, Navas E, Hernáez I, Sánchez J. Automatic emotion recognition using prosodic parameters. Proceedings of the 9th European Conference on Speech Communication and Technology; 2005 Sep; Lisboa, Portugal. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.149.4890&rep=rep1&type=pdf.

Lyons J. python_speech_features. Documentation release 0.1.0. 2017. https://python-speech-features.readthedocs.io/en/latest/.

Miller CA, Parasuraman R. Designing for flexible interaction between humans and automation: delegation interfaces for supervisory control. Human Factors. 2007;49(1):57–75. https://doi.org/10.1518/001872007779598037.

Mirsamadi S, Barsoum E, Zhang C. Automatic speech emotion recognition using recurrent neural networks with local attention. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing; 2017 Mar; New Orleans, LA. https://doi.org/10.1109/ICASSP.2017.7952552.

Neubauer C, Chollet M, Mozgai S, Dennison M, Khooshabeh P, Scherer S. The relationship between task-induced stress, vocal changes, and physiological state during a dyadic team task. Proceedings of the 19th ACM International Conference on Multimodal Interaction; 2017 Jan; Glasgow, Scotland. https://doi.org/10.1145/3136755.3136804.

Neumann M, Vu NT. Attentive convolutional neural network based speech emotion recognition: a study on the impact of input features, signal length, and acted speech. 2017 June. http://arxiv.org/abs/1706.00612.

Nevatia Y, Stoyanov T, Rathnam R, Pfingsthorn M, Markov S, Ambrus R, Birk A. Augmented autonomy: improving human-robot team performance in urban search and rescue. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems; 2008 Sep; Nice France. https://doi.org/10.1109/IROS.2008.4651034.

Parasuraman R, Mouloua M, Molloy R. Effects of adaptive task allocation on monitoring of automated systems. Human Factors. 1996;38(4):665–679. https://doi.org/10.1518/001872096778827279.

Parasuraman R, Sheridan TB, Wickens, CD. A model for types and levels of human interaction with automation. IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans. 2000;30(3):286–297. https://doi.org/10.1109/3468.844354.

Parasuraman R, Cosenzo KA, De Visser E. Adaptive automation for human supervision of multiple uninhabited vehicles: effects on change detection, situation awareness, and mental workload. Military Psychology. 2009;21(2). https://doi.org/10.1080/08995600902768800.

Perelman BS, Wright JL, Lieberman GA, Lakhmani S. Human–autonomy teaming essential research program. Project 2: transparent multimodal crew interface designs. Technical note 2: transparency in mobility planning. DEVCOM Army Research Laboratory; 2020 Jan. Report No.: ARL-TN-1004.

Phillips E, Ososky S, Grove J, Jentsch F. From tools to teammates: toward the development of appropriate mental models for intelligent robots. Proceedings of the Human Factors and Ergonomics Society; 2011 Sep. https://doi.org/10.1177/1071181311551310.

Ramakrishnan S. Recognition of emotion from speech : a review. In: Speech enhancement, modeling and recognition – algorithms and applications. IntechOpen; 2012 Mar. https://www.intechopen.com/books/speech-enhancement-modeling-and-recognition-algorithms-and-applications.

Sacks H, Schegloff EA, Jefferson G. A simplest systematics for the organization of turn-taking for conversation. Language. 1974;50(4):696–735.

Salas E, Sims DE, Burke CS. Is there a “big five” in teamwork? Small Group Research. 2005;36(5). https://doi.org/10.1177/1046496405277134.

Sauer J, Kao CS, Wastell D. A comparison of adaptive and adaptable automation under different levels of environmental stress. Ergonomics. 2012;55(8):840–853. https://doi.org/10.1080/00140139.2012.676673.

Schaefer KE, Brewer RW, Baker AL, Pursel R, Gipson B, Ratka S, Giacchi J, Cerame E, Pirozzo K. US Army wingman joint capability technology demonstration (JCTD): initial Soldier and Marine feedback on manned–unmanned gunnery operations. Army Research Laboratory; 2018 Mar. Report No.: ARL-TR-8663.

Schmidhuber J. Learning in neural networks: an overview. Neural Networks. 2015 Jan;61:85–117. https://doi.org/10.1016/j.neunet.2014.09.003.

Schuller B, Rigoll G, Lang M. Hidden Markov model-based speech emotion recognition. Proceedings of the IEEE International Conference on Multimedia and Expo; 2003 July; Baltimore, MD. https://doi.org.10.1109/ICME .2003.1220939.

Sheridan TB. Adaptive automation, level of automation, allocation authority, supervisory control, and adaptive control: distinctions and modes of adaptation. IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans; 2011;41(4). https://doi.org/10.1109/TSMCA.2010 .2093888.

South T. From unmanned teaming to new capabilities, the Army's next generation combat vehicle will be a leap forward. Army Times. 2019 Oct 20. https://www.armytimes.com/news/your-army/2019/10/20/from-unmanned-teaming-to-new-capabilities-the-armys-next-generation-combat-vehicle-will-be-a-leap-forward/.

Stuhlsatz A, Meyer C, Eyben F, Zielke T, Meier G, Schuller B. Deep neural networks for acoustic emotion recognition: raising the benchmarks. Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing; 2011 May; Prague, Czech Republic. https://doi.org/10.1109/ICASSP.2011.5947651.

Swain M, Routray A, Kabisatpathy P. Databases, features and classifiers for speech emotion recognition: a review. International Journal of Speech Technology. 2018;21(1). https://doi.org/10.1007/s10772-018-9491-z.

Thompson LL. Team cohesion and trust. In: Making the team: a guide for managers. 6th ed. Pearson; 2016. p. 82–108.

Tiferes J, Hussein AA, Bisantz A, Kozlowski JD, Sharif MA, Winder NM, Ahmad N, Allers J, Cavuoto L, Guru KA. The loud surgeon behind the console: understanding team activities during robot-assisted surgery. Journal of Surgical Edrcation. 2016;73(3). https://doi.org/10.1016/j.jsurg.2015.12.009.

van Rossum G, Drake FL. Python 3 reference manual. CreateSpace Independent Publishing Platform; 2009 Mar 20.

Vergyri D, Knoth B, Shriberg E, Mitra V, McLaren M, Ferrer L, Garcia P, Marmar C. Speech-based assessment of PTSD in a military population using diverse feature classes. Proceedings of the Annual Conference of the International Speech Communication Association; 2015 Sep; Dresden, Germany. p. 3729–3733.

Ververidis D, Kotropoulos C. A review of emotional speech databases. 2003. p. 560–574. http://delab.csd.auth.gr/bci1/Panhellenic/560ververidis.pdf.

Ververidis D, Kotropoulos C. Emotional speech recognition: resources, features, and methods. Speech Communication. 2006;48(9):1162–1181. https://www.sciencedirect.com/science/article/abs/pii/S0167639306000422.

Vidrascu L, Devillers L. Real-life emotion representation and detection in call centers data. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics). Springer; 2005. p. 739–746. https://link.springer.com/chapter/10.1007/11573548_95.

Vogt T, André E. Feature sets for acted and spontaneous speech in view of automatic emotion recognition. Proceedings of the IEEE International Conference on Multimedia and Expo; 2005 July; Amsterdam, Netherlands. https://doi.org/10.1109/ICME.2005.1521463.

Vogt T, André E, Wagner J. Automatic recognition of emotions from speech: a review of the literature and recommendations for practical realisation. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics). Springer; 2008. p. 75–91. https://link.springer.com/chapter/10.1007/978-3-540-85099-1_7.

Waldinger RJ, Hauser ST, Schulz MS, Allen JP, Crowell JA. Reading others' emotions: the role of intuitive judgments in predicting marital satisfaction, quality, and stability. Journal of Family Psychology. 2004;18(1):58–71. https://psycnet.apa.org/record/2004-11293-006.

Yacoub SM, Simske SJ, Lin X, Burns, J. Recognition of emotions in interactive voice response systems. Proceedings of the 8th European Conference on Speech Communication and Technology; 2003 Sep; Geneva, Switzerland. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.4915&rep=rep1&type=pdf.

Zhang S, Zhang S, Huang T, Gao W. Speech emotion recognition using deep convolutional neural network and discriminant temporal pyramid matching. IEEE Transactions on Multimedia. 2018;20(6). https://doi.org/10.1109/TMM.2017.2766843.

Appendix A. Code Sample (zscore_WMtrin.py)

This appendix appears in its original form, without editorial change.

```
"""
zscore_WMtrin.py
Created on Thu Jul 16 10:07:32 2020

@author: scharine

python 3.7.4 TF 2
This code reads in the wave files and normalizes them and exports
it to a pickle file. This file is then used in
ExtractMel_WMtrin.py
"""


import wave
import numpy as np
import python_speech_features as ps #James Lyons
import os
from six.moves import cPickle as cPickle #Deprecated - but it
makes a pickle file out of the data

eps = 1e-5
def wgn(x, snr):
    snr = 10**(snr/10.0)
    xpower = np.sum(x**2)/len(x)
    npower = xpower / snr
    return np.random.randn(len(x)) * np.sqrt(npower)
def
getlogspec(signal,samplerate=16000,winlen=0.02,winstep=0.01,

nfilt=26,nfft=399,lowfreq=0,highfreq=None,preemph=0.97,
               winfunc=lambda x:np.ones((x,))):
    highfreq= highfreq or samplerate/2
    signal = ps.sigproc.preemphasis(signal,preemph)
    frames  =  ps.sigproc.framesig(signal,  winlen*samplerate,
winstep*samplerate, winfunc)
    pspec = ps.sigproc.logpowspec(frames,nfft)
    return pspec
def read_file(filename):
    file = wave.open(filename,'r')
    params = file.getparams()
    nchannels, sampwidth, framerate, wav_length = params[:4]
    str_data = file.readframes(wav_length)
    wavedata = np.frombuffer(str_data, dtype = np.short)
    #wavedata  =  np.float(wavedata*1.0/max(abs(wavedata)))    #
normalization)
    time = np.arange(0,wav_length) * (1.0/framerate)
    file.close()
    return wavedata, time, framerate
def dense_to_one_hot(labels_dense, num_classes):
  """Convert class labels from scalars to one-hot vectors."""
  num_labels = labels_dense.shape[0]
  index_offset = np.arange(num_labels) * num_classes
  labels_one_hot = np.zeros((num_labels, num_classes))
  labels_one_hot.flat[index_offset + labels_dense.ravel()] = 1
```

```
  return labels_one_hot


def zscore(data,mean,std):
    shape = np.array(data.shape,dtype = np.int32)
    for i in range(shape[0]):
        data[i,:,:,0] = (data[i,:,:,0]-mean)/(std)
    return data

def normalization(data):
    '''
    #apply zscore
    train_data = zscore(train_data,mean,std)
    test_data = zscore(test_data,mean,std)
    '''
    mean = np.mean(data,axis=0)#axis=0
    std = np.std(data,axis=0)
    data = (data-mean)/std
    return data
def mapminmax(data):
    shape = np.array(data.shape,dtype = np.int32)
    for i in range(shape[0]):
        min = np.min(data[i,:,:,0])
        max = np.max(data[i,:,:,0])
        data[i,:,:,0] = (data[i,:,:,0] - min)/((max - min)+eps)
    return data
def generate_label(emotion,classnum):
    label = -1
    if(emotion == 'nsp'):
        label = 0
    elif(emotion == 'neu'):
        label = 1
    else:
        label = 2 #'eng'
    return label


def read_CASIA():

    train_num = 10278
    filter_num = 40
    rootdir = 'C://Users/angelique.scharine/Documents/wingman-
python/tests'
    traindata1                                                  =
np.empty((train_num*300,filter_num),dtype=np.float32)
    traindata2                                                  =
np.empty((train_num*300,filter_num),dtype=np.float32)
    traindata3                                                  =
np.empty((train_num*300,filter_num),dtype=np.float32)
    train_num = 0
    num_neu = 0
    num_eng = 0
    num_nsp = 0
```

```
for test in os.listdir(rootdir):
    if(test[0] == 't'):
        counter_nsp = 0
        counter_neu = 0
        counter_eng = 0

        #print(rootdir)
        #print(test)
        sub_dir = rootdir+'/'+test+'/wav/'
        #print(sub_dir)
        emoevl = rootdir+'/'+test+'/EmoEvaluation'
        #print(emoevl)
        for part in os.listdir(emoevl):
            pt = part.split('.')
            part2 = pt[0]
            #print(part2)
            if(part[0] == 'p'):
                emotfile = emoevl+'/'+part
                #print(emotfile)
                emot_map = {}
                with open(emotfile,'r') as emot_to_read:
                    while True:
                        line = emot_to_read.readline()
                        if not line:
                            break
                        if(line[0] == '['):
                            t = line.split()
                            emot_map[t[3]] = t[4]
                            #print(emot_map)
                            #print(t[3])
                            #print(t[4])


                file_dir = sub_dir+part2
                #print(file_dir)
                for filename in os.listdir(file_dir):
                    wavname = file_dir+'/'+filename
                    #print (wavname)
                    s = wavname.split("/")
                    #print(s)
                    wavtag = s[10]
                    wavtag = os.path.splitext(wavtag)[0]
                    #print(wavtag)
                    emotion = emot_map[wavtag]
                    #print(emotion)

                    if emotion == 'neu':
                        num_neu = num_neu + 1
                        counter_neu = counter_neu + 1
                    elif emotion == 'nsp':
                        num_nsp = num_nsp + 1
                        counter_nsp = counter_nsp + 1
                    else:
```

```
                            num_eng = num_eng + 1
                            counter_eng = counter_eng + 1
                        #print(traindata1.shape[0])

                        if(train_num+1)*300                 <=
traindata1.shape[0]:
                            #print(train_num)
                            if(emotion in ['neu','nsp','eng']):
                                #print(wavname)
                                data,     time,     rate     =
read_file(wavname)
                                mel_spec                    =
ps.logfbank(data,rate,nfilt = filter_num)
                                delta1 = ps.delta(mel_spec, 2)
                                delta2 = ps.delta(delta1, 2)

                                time = mel_spec.shape[0]
                                if(time <= 300):
                                    part = mel_spec
                                    delta11 = delta1
                                    delta21 = delta2
                                    part  =  np.pad(part,((0,300
- part.shape[0]),(0,0)),'constant',constant_values = 0)
                                    delta11                 =
np.pad(delta11,((0,300                                      -
delta11.shape[0]),(0,0)),'constant',constant_values = 0)
                                    delta21                 =
np.pad(delta21,((0,300                                      -
delta21.shape[0]),(0,0)),'constant',constant_values = 0)

traindata1[train_num*300:(train_num+1)*300] = part

traindata2[train_num*300:(train_num+1)*300] = delta11

traindata3[train_num*300:(train_num+1)*300] = delta21

                                    em                      =
generate_label(emotion,3)
                                    #print(em)
                                    train_num = train_num + 1
                                else:

                                    for i in range(2):
                                        if(i == 0):
                                            begin = 0
                                            end = begin + 300
                                        else:
                                            begin = time - 300
                                            end = time

                                        part                =
mel_spec[begin:end,:]
                                        delta11             =
delta1[begin:end,:]
```

```
                                        delta21                =
delta2[begin:end,:]

traindata1[train_num*300:(train_num+1)*300] = part

traindata2[train_num*300:(train_num+1)*300] = delta11

traindata3[train_num*300:(train_num+1)*300] = delta21
                                         train_num = train_num +
1
                                    else:
                                        frames   =   divmod(time-
300,100)[0] + 1
                                        if  (train_num  +frames-
1)*300 >= traindata1.shape[0]:
                                            frames             =
int(traindata1.shape[0]/300 - train_num)
                                            #print(frames)
                                        for i in range(frames):
                                            begin = 100*i
                                            end = begin + 300
                                            part               =
mel_spec[begin:end,:]
                                            delta11            =
delta1[begin:end,:]
                                            delta21            =
delta2[begin:end,:]

#print(traindata1.shape[0])

#print(train_num*300)

traindata1[train_num*300:(train_num+1)*300] = part

traindata2[train_num*300:(train_num+1)*300] = delta11

traindata3[train_num*300:(train_num+1)*300] = delta21
                                            train_num          =
train_num + 1

                        else:
                            #print(file_dir)
                            #print(traindata1.shape[0])
                            #print(train_num)
                            #print(num_neu)
                            #print(num_eng)
                            #print(num_nsp)
                            pass
                    else:
                        pass
        #print(train_num) #10278
#       print(test,'number nspnonspeech = ',counter_nsp) #779
#       print(test,'number neutral = ',counter_neu) #381
#       print(test,'number engaged = ',counter_eng) #424
```

```
#         test2 number neutral =  114
#         test2 number engaged =  164
#         test3 number nonspeech =  205
#         test3 number neutral =  128
#         test3 number engaged =  82
#         test5 number nonspeech =  184
#         test5 number neutral =  84
#         test5 number engaged =  100
#         test6 number nonspeech =  120
#         test6 number neutral =  55
#         test6 number engaged =  78


        mean1 = np.mean(traindata1,axis=0)#axis=0
        std1 = np.std(traindata1,axis=0)
        mean2 = np.mean(traindata2,axis=0)#axis=0
        std2 = np.std(traindata2,axis=0)
        mean3 = np.mean(traindata3,axis=0)#axis=0
        std3 = np.std(traindata3,axis=0)
        output = './zscore_Trin'+str(filter_num)+'.pkl'
        #output = './IEMOCAP'+str(m)+'_'+str(filter_num)+'.pkl'
        #print('num_neu = ', num_neu)
        #print('num_ang = ', num_ang)
        f=open(output,'wb')
        cPickle.dump((mean1,std1,mean2,std2,mean3,std3),f)
        f.close()
    return



if __name__=='__main__':
    read_CASIA()
    #print "test_num:", test_num
    #print "train_num:", train_num
#     n = wgn(x, 6)
#     xn = x+n
```

Appendix B. Code Sample (ExtractMel_WMtrin.py)

This appendix appears in its original form, without editorial change.

```
# -*- coding: utf-8 -*-
"""
ExtractMel_WMtrin.py
Created on Thu Jul 16 20:32:28 2020

@author: Scharine
python 3.7.4 TF 2

This code takes the normalized wave file information from
zscore_WMtrin.py and computes the MFCC and derivative
information. It is using James Lyon's python_speech_features
code.
"""

import wave
import numpy as np
import python_speech_features as ps
import os
from six.moves import cPickle as cPickle #Deprecated - but it
makes a pickle file out of the data

eps = 1e-5
def wgn(x, snr):
    snr = 10**(snr/10.0)
    xpower = np.sum(x**2)/len(x)
    npower = xpower / snr
    return np.random.randn(len(x)) * np.sqrt(npower)
def
getlogspec(signal,samplerate=16000,winlen=0.02,winstep=0.01,

nfilt=26,nfft=399,lowfreq=0,highfreq=None,preemph=0.97,
               winfunc=lambda x:np.ones((x,))):
    highfreq= highfreq or samplerate/2
    signal = ps.sigproc.preemphasis(signal,preemph)
    frames  =  ps.sigproc.framesig(signal,  winlen*samplerate,
winstep*samplerate, winfunc)
    pspec = ps.sigproc.logpowspec(frames,nfft)
    return pspec
def read_file(filename):
    file = wave.open(filename,'r')
    params = file.getparams()
    nchannels, sampwidth, framerate, wav_length = params[:4]
    str_data = file.readframes(wav_length)
    wavedata = np.frombuffer(str_data, dtype = np.short)
    #wavedata  =  np.float(wavedata*1.0/max(abs(wavedata)))    #
normalization)
    time = np.arange(0,wav_length) * (1.0/framerate)
    file.close()
    return wavedata, time, framerate
def dense_to_one_hot(labels_dense, num_classes):
  """Convert class labels from scalars to one-hot vectors."""
  num_labels = labels_dense.shape[0]
  index_offset = np.arange(num_labels) * num_classes
  labels_one_hot = np.zeros((num_labels, num_classes))
```

```
  labels_one_hot.flat[index_offset + labels_dense.ravel()] = 1
  return labels_one_hot

def zscore(data,mean,std):
    shape = np.array(data.shape,dtype = np.int32)
    for i in range(shape[0]):
        data[i,:,:,0] = (data[i,:,:,0]-mean)/(std)
    return data

def normalization(data):
    '''
    #apply zscore
    wingman_data = zscore(wingman_data,mean,std)
    test_data = zscore(test_data,mean,std)
    '''
    mean = np.mean(data,axis=0)
    std = np.std(data,axis=0)
    data = (data-mean)/std
    return data
def mapminmax(data):
    shape = np.array(data.shape,dtype = np.int32)
    for i in range(shape[0]):
        min = np.min(data[i,:,:,0])
        max = np.max(data[i,:,:,0])
        data[i,:,:,0] = (data[i,:,:,0] - min)/((max - min)+eps)
    return data
def generate_label(emotion,classnum):
    label = -1
    if(emotion == 'nsp'):
        label = 0
    elif(emotion == 'neu'):
        label = 1
    else:
        label = 2 #engaged
    return label
def load_data():
    f = open('./zscore_Trin40.pkl','rb') #pickle file obtained
from zsore_WMtrin.py
    mean1,std1,mean2,std2,mean3,std3 = cPickle.load(f)
    return mean1,std1,mean2,std2,mean3,std3


def read_IEMOCAP(): # based on earlier code that read IEMOCAP
files
    eps = 1e-5
    tnum = 394 #the number of test utterance
    vnum = 394
    train_num = 1811
    test_num = 538 #the number of test 2s segments - some
utterances are a single segment, others are longer
    valid_num = 633
    #wingman_num = 1708
    wingman_num = 2589
    filter_num = 40
```

```
    pernums_test = np.arange(tnum)#remember each utterance
contain how many segments
    pernums_valid = np.arange(vnum)
    rootdir = 'C://Users/angelique.scharine/Documents/wingman-
python/tests'

    mean1,std1,mean2,std2,mean3,std3 = load_data()

    #1584
    nspnum = 1379#0
    neunum = 577#1
    engnum = 633#2

    #pernum = 1200 #np.min([nspnum,neunum, engnum])

    wingman_label = np.empty((wingman_num,1), dtype = np.int8)
    train_label = np.empty((train_num,1), dtype = np.int8)
    test_label = np.empty((tnum,1), dtype = np.int8)
    valid_label = np.empty((vnum,1), dtype = np.int8)
    Test_label = np.empty((test_num,1), dtype = np.int8)
    Valid_label = np.empty((valid_num,1), dtype = np.int8)
    wingman_data                                                =
np.empty((wingman_num,300,filter_num,3),dtype = np.float32)
    train_data = np.empty((train_num,300,filter_num,3), dtype =
np.float32)
    test_data  =  np.empty((test_num,300,filter_num,3),dtype  =
np.float32)
    valid_data = np.empty((valid_num,300,filter_num,3),dtype =
np.float32)

    tnum = 0
    vnum = 0
    wingman_num = 0
    wingman_emt = {'nsp':0,'neu':0,'eng':0}
    train_emt = {'nsp':965,'neu':403,'eng':443}
    test_emt = {'nsp':207,'neu':87,'eng':95}
    valid_emt = {'nsp':207,'neu':87,'eng':87}
    for test in os.listdir(rootdir):
        if(test[0] == 't'):
            sub_dir = rootdir+'/'+test+'/wav/'
            emoevl = rootdir+'/'+test+'/EmoEvaluation'
            for part in os.listdir(emoevl):
                pt = part.split('.')
                part2 = pt[0]
                #print(part2)
                if(part[0] == 'p'):
                    emotfile = emoevl+'/'+part
                    #emotfile = open(emotdir)
                    emot_map = {}
                    with open(emotfile,'r') as emot_to_read:
                        while True:
                            line = emot_to_read.readline()
                            if not line:
                                break
```

```
                        if(line[0] == '['):
                            t = line.split()
                            emot_map[t[3]] = t[4]


                    file_dir = sub_dir+part2

                    for filename in os.listdir(file_dir):
                        wavname = file_dir+'/'+filename
                        s = wavname.split("/")
                        wavtag = s[10]
                        wavtag = os.path.splitext(wavtag)[0]
                        emotion = emot_map[wavtag]
                        #print(emotion)
                        count = 0
                        if(emotion in ['nsp','neu','eng']):
                            data,       time,       rate       =
read_file(wavname)
                            mel_spec                           =
ps.logfbank(data,rate,nfilt = filter_num)
                            delta1 = ps.delta(mel_spec, 2)
                            delta2 = ps.delta(delta1, 2)
                            time = mel_spec.shape[0]
                            count = count + 1
                            #all_data
                            if(time <= 300):
                                 partl = mel_spec
                                 delta11 = delta1
                                 delta21 = delta2
                                 partl = np.pad(partl,((0,300 -
partl.shape[0]),(0,0)),'constant',constant_values = 0)
                                 delta11                       =
np.pad(delta11,((0,300                                         -
delta11.shape[0]),(0,0)),'constant',constant_values = 0)
                                 delta21                       =
np.pad(delta21,((0,300                                         -
delta21.shape[0]),(0,0)),'constant',constant_values = 0)

wingman_data[wingman_num,:,:,0] = (partl - mean1)/(std1+eps)

wingman_data[wingman_num,:,:,1] = (delta11 - mean2)/(std2+eps)

wingman_data[wingman_num,:,:,2] = (delta21 - mean3)/(std3+eps)
                                 em = generate_label(emotion,3)
                                 wingman_label[wingman_num]   =
em
                                 wingman_emt[emotion]         =
wingman_emt[emotion] + 1
                                 wingman_num = wingman_num + 1
                                 #print('wingman_num    =    ',
wingman_num)
                                 #print(emotion)
                            else:
```

```
                                    for i in range(2):
                                        if(i == 0):
                                            begin = 0
                                            end = begin + 300
                                        else:
                                            begin = time - 300
                                            end = time

                                        partl                    =
mel_spec[begin:end,:]
                                        delta11                  =
delta1[begin:end,:]
                                        delta21                  =
delta2[begin:end,:]

wingman_data[wingman_num,:,:,0] = (partl -mean1)/(std1+eps)

wingman_data[wingman_num,:,:,1] = (delta11 - mean2)/(std2+eps)

wingman_data[wingman_num,:,:,2] = (delta21 - mean3)/(std3+eps)
                                        em                       =
generate_label(emotion,3)

wingman_label[wingman_num] = em
                                        wingman_emt[emotion]     =
wingman_emt[emotion] + 1
                                        wingman_num  =  wingman_num
+ 1
                                        #print('wingman_num  =  ',
wingman_num)
                                        #print(emotion)


                            else:
                                pass
                        else:
                            pass


#    print('wingman_num = ', wingman_num)
#    print('test_num = ', test_num)
#    print('valid_num = ', valid_num)
#    print('tnum = ', tnum)
#    print('vnum = ',vnum)
#    print(wingman_emt)
#    print(test_emt)
#    print(valid_emt)


    nsp_index = np.arange(nspnum) #empty array for nsp segments
    neu_index = np.arange(neunum) #empty array for neu segments
    eng_index = np.arange(engnum) #empty array for eng segments
    q0 = 0
    n1 = 0
```

```
    e2 = 0
    for l in range(wingman_num): #fills the arrays
        #print(wingman_label[l])
        if (wingman_label[l] == 0):
            nsp_index[q0] = l
            q0 = q0 + 1
            #print(nsp_index[q0])
        elif (wingman_label[l] == 1):
            neu_index[n1] = l
            n1 = n1 + 1
            #print(neu_index[n1])
        else:
            eng_index[e2] = l
            e2 = e2 + 1
            #print(eng_index[e2])
#    print(nsp_index.shape)
#    print(neu_index.shape)
#    print(eng_index.shape)
    for m in range(1):
        np.random.shuffle(nsp_index)   #randomizes   the   nsp
segments
        np.random.shuffle(neu_index)   #randomizes   the   neu
segments
        np.random.shuffle(eng_index)   #randomizes   the   eng
segments
        #define emotional array
        nsp_label = np.empty((1379,1), dtype = np.int8)
        eng_label = np.empty((633,1), dtype = np.int8)
        neu_label = np.empty((577,1), dtype = np.int8)
        nsp_data  =  np.empty((1379,300,filter_num,3),dtype  =
np.float32)
        eng_data  =  np.empty((633,300,filter_num,3),dtype  =
np.float32)
        neu_data  =  np.empty((577,300,filter_num,3),dtype  =
np.float32)

        nsp_data   =   wingman_data[nsp_index[0:1379]].copy()
#fills array called wingman_data with segments
        nsp_label = wingman_label[nsp_index[0:1379]].copy()
        eng_data = wingman_data[eng_index[0:633]].copy()
        eng_label = wingman_label[eng_index[0:633]].copy()
        neu_data = wingman_data[neu_index[0:577]].copy()
        neu_label = wingman_label[neu_index[0:577]].copy()
        train_num = 1811

        train_label = np.empty((train_num,1), dtype = np.int8)
        train_data                                          =
np.empty((train_num,300,filter_num,3),dtype = np.float32)
        train_data[0:965] = nsp_data[0:965] #assigns first 70%
of segments to the training data set
        train_label[0:965] = nsp_label[0:965] #assigns labels
        train_data[965:1368] = neu_data[0:403]
        train_label[965:1368] = neu_label[0:403]
        train_data[1368:1811] = eng_data[0:443]
```

```
        train_label[1368:1811] = eng_label[0:443]

        test_label = np.empty((test_num,1),dtype = np.int8)
        test_data = np.empty((test_num, 300, filter_num, 3),
dtype = np.float32)
        test_data[0:207] = nsp_data[965:1172] #assigns 15% to
test set
        test_label[0:207] = nsp_label[965:1172]
        test_data[207:294] = neu_data[403:490]
        test_label[207:294] = neu_label[403:490]
        test_data[294:389] = eng_data[443:538]
        test_label[294:389] = eng_label[443:538]

        valid_data = np.empty((valid_num,300, filter_num, 3),
dtype = np.float32)
        valid_label= np.empty((valid_num, 1), dtype = np.int8)
        valid_data[0:207] = nsp_data[1172:1379] #assigns 15% to
validation set
        valid_label[0:207] = nsp_label[1172:1379]
        valid_data[207:294] = neu_data[490:577]
        valid_label[207:294] = neu_label[490:577]
        valid_data[294:389] = eng_data[538:633]
        valid_label[294:389] = eng_label[538:633]

        arr = np.arange(train_num)
        np.random.shuffle(arr)
        train_data = train_data[arr[0:]]
        train_label = train_label[arr[0:]]
        arr = np.arange(test_num)
        np.random.shuffle(arr)
        test_data = test_data[arr[0:]]
        test_label = test_label[arr[0:]]
        arr = np.arange(valid_num)
        np.random.shuffle(arr)
        valid_data = valid_data[arr[0:]]
        valid_label = valid_label[arr[0:]]
        print(train_label.shape)
        print(train_data.shape)
        print(train_emt)
        print(test_emt)
        print(valid_emt)
        #print(test_data)
        #print(test_label)
        #print(valid_data)
        #print(valid_label)
        #print(train_data)
        #print(train_label)
        #print(test_label[0:500,:])
        # f=open('./CASIA_40_delta.pkl','wb')
        # output = './IEMOCAP40.pkl'
        output = './WM_TRIN.pkl' #datafile used in the CRNN
model
        f=open(output,'wb')
```

```
cPickle.dump((wingman_data,wingman_label,test_data,test_label,
valid_data,valid_label,Valid_label,Test_label,pernums_test,per
nums_valid),f)
        f.close()
    return



if __name__=='__main__':
    read_IEMOCAP()
    #print "test_num:", test_num
    #print "wingman_num:", wingman_num
#    n = wgn(x, 6)
#    xn = x+n
```

Appendix C. Code Sample (SpeechEmotClassifierWMtrin.py)

This appendix appears in its original form, without editorial change.

```
# -*- coding: utf-8 -*-
""" SpeechEmotClassifierWMtrin.py

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1Swyz1abkYJrnTu3sHQ6hW
9waMIszkd1s

This is the CRNN model that classifies wave files containing
speech as either non-speech (engine noise only), neutral, or
engaged.
It takes as input, a pickle file that contains MFCCs and their
derivatives of the wave files.

python 3.7.4 TF 2
"""

from google.colab import drive
drive.mount('/content/gdrive')

import os
from six.moves import cPickle as cPickle
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
print(tf.__version__)

fname = '/content/gdrive/My Drive/Colab Notebooks/WM_TRIN.pkl'

with open(fname, 'rb') as f:

train_data,train_label,test_data,test_label,valid_data,valid_l
abel,Valid_label,Test_label,pernums_test,pernums_valid        =
pd.read_pickle(f)

train_data.shape



model = models.Sequential()
model.add(layers.Conv2D(128,    (5,    3),    activation='relu',
input_shape=(300, 40, 3)))
model.add(layers.BatchNormalization())
model.add(layers.LeakyReLU(.1))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.2))
model.add(layers.Conv2D(256, (5, 3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.LeakyReLU(.1))
```

```
model.add(layers.Dropout(0.2))
model.add(layers.Conv2D(256, (5, 3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.LeakyReLU(.1))
model.add(layers.Dropout(0.2))
model.add(layers.Conv2D(256, (5, 3), activation='relu'))
model.add(layers.BatchNormalization())
model.add(layers.LeakyReLU(.1))
model.add(layers.Dropout(0.2))
model.add(layers.Reshape((256,       1768),input_shape       =
(136,13,256)))
model.add(layers.Bidirectional(layers.LSTM(128,
return_sequences=True),
                               merge_mode='concat', input_shape
= (None, 256, 1152)
                               ))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(3))

model.summary()

model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
=True),
              metrics=['accuracy'])

history   =   model.fit(train_data,   train_label,   epochs=20,
validation_data=(test_data, test_label))

plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'],       label       =
'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(valid_data,  valid_label,
verbose=2)

print(test_acc)
```

List of Symbols, Abbreviations, and Acronyms

3-D	three-dimensional
AAA	adaptive autonomous assistance
BLSTM	bidirectional long short-term memory
C2	command and control
CNN	convolutional neural network
CRNN	convolutional recurrent neural network
DEVCOM	US Army Combat Capabilities Development Command
DIDEA	detect, identify, decide, engage, and assess
DNN	deep neural network
eng	engaged
HAT	Human–Autonomy Teaming
HMMWV	high-mobility multipurpose wheeled vehicle
IEMOCAP	Interactive Emotional Dyadic Motion Capture (database)
JCTD	Joint Capability Technology Demonstration
LSTM	long short-term memory
MET-D	mission-enabling technologies demonstrator
MFCC	Mel-frequency cepstrum coefficient
ML	machine learning
neu	neutral
NGCV	Next-Generation Combat Vehicle
NN	neural network
nsp	nonspeech
RNN	recurrent neural network
SER	speech emotion recognition

1 (PDF) DEFENSE TECHNICAL INFORMATION CTR DTIC OCA

1 (PDF) DEVCOM ARL FCDD RLD DCI TECH LIB

1 (PDF) DEVCOM ARL FCDD RLH B T DAVIS BLDG 5400 RM C242 REDSTONE ARSENAL AL 35898-7290

1 (PDF) DEVCOM ARL FCDD HSI J THOMAS 6662 GUNNER CIRCLE ABERDEEN PROVING GROUND MD 21005-5201

1 (PDF) USAF 711 HPW 711 HPW/RH K GEISS 2698 G ST BLDG 190 WRIGHT PATTERSON AFB OH 45433-7604

1 (PDF) USN ONR ONR CODE 341 J TANGNEY 875 N RANDOLPH STREET BLDG 87 ARLINGTON VA 22203-1986

1 (PDF) USA NSRDEC RDNS D D TAMILIO 10 GENERAL GREENE AVE NATICK MA 01760-2642

1 (PDF) OSD OUSD ATL HPT&B B PETRO 4800 MARK CENTER DRIVE SUITE 17E08 ALEXANDRIA VA 22350

3 (PDF) USCENTCOM A LESTER G TIGHE A VAID

3 (PDF) GROUND VEHICLES SYSTEM CENTER RTI GVR T MICHALIK K BRIGGS J PALOMINO

1 (PDF) NGCV CFT C WALLACE

ABERDEEN PROVING GROUND

18 (PDF) DEVCOM ARL FCDD RLH J LANE Y CHEN P FRANASZCZUK K MCDOWELL K OIE A MARATHE FCDD RLH F J GASTON (A) FCDD RLH FA A DECOSTANZA J CANADY FCDD RLH FB D BOOTHE (A) FCDD RLH FC K COX (A) FCDD RLH FD A FOOTS A SCHARINE FCDD RLH FE D HEADLEY FCDD RLH P A EVANS FCDD RLV J RIDDICK FCDD RLV A K E SCHAEFER-LAY R W BREWER