

# Emotion detection from sound. Project report

December 28, 2019

by Alena Churakova

## 1 Introduction

### 1.1 Domain background

Much of the natural language understanding by a machine is based on study of text. This project focuses on recognition of emotions from the sound of human speech. Researchers recognize the importance of teaching machines to recognize human emotions, which can be used to adapt machine's action appropriately <sup>1</sup>. The rise of virtual personal assistants shows people willingness to interact with machines by voice. Analysis of tone in addition to the text information has a potential to make the interactions more natural, pleasant and effective.

### 1.2 Problem statement

A sentence with a neutral meaning, e.g. "This is a dog", carries factual content. Depending on the context and a person's attitude. The same sentence can be pronounced with happiness in the voice by a child who was secretly hoping their parents get a puppy a birthday present. A burglar, on the contrary, would likelier say it with fear.

The task is to recognize emotions from the audial signals of human speech, without involvement of text analysis. A model developed in this project can be useful in an application where a machine changes its action based on emotion detection, e.g. encourage the conversation partner when detecting sadness.

### 1.3 Evaluation metrics

In a described example scenario of changing machine's actions based on the detected emotion, an accurate prediction brings value of appropriate reaction by a machine. This motivates to use accuracy as an evaluation metric from the application/business perspective. Given a balanced dataset, this metric seems appropriate from a methodological perspective as well. For a multiclass classification, accuracy is defined as a ratio of correctly classified cases to all cases.

---

<sup>1</sup>Maghilnan S and Rajesh Kumar M. Sentiment Analysis on Speaker Specific Speech Data <https://arxiv.org/pdf/1802.06209.pdf>

## 2 Development

### 2.1 Data exploration

As outlined in the project proposal, the Toronto emotional speech set (TESS)<sup>2</sup> was used. It consists of short audiofiles (.wav format) recorded by two actors of different ages. Labels in the analyzed dataset name emotions. Having heard multiple audio files, it became clear that a human would misclassify some of them. This happens because labels depend on interpretation of the speaker and overall, emotions labelling is difficult for people, let alone doing it from the voice.

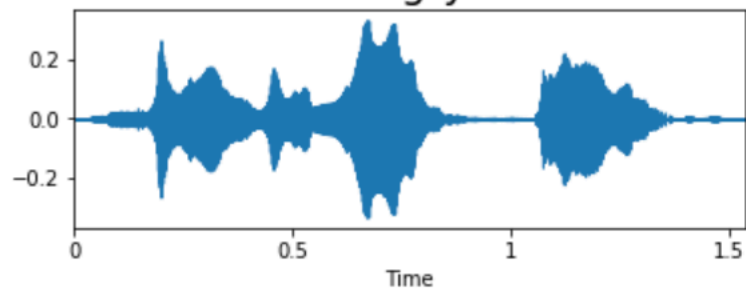
This project concentrated on a few expressions — sad, happy, fear, angry and neutral. The dataset is balanced and the total number of files in each category is 400.

Visual inspection of audio represented as a time-series with the amplitude on the y-axis does not reveal large differences between emotions. This is likely be due to fact that audio files have a strict pattern 'Say the word \_\_'.

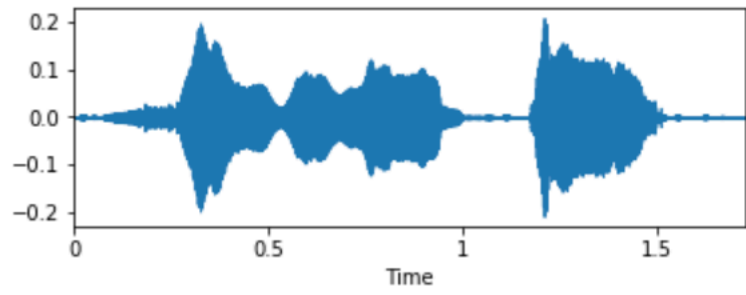
---

<sup>2</sup>Dupuis, Kate and Pichora-Fuller, M. Kathleen (2010). Toronto emotional speech set (TESS) <https://tspace.library.utoronto.ca/handle/1807/24487>

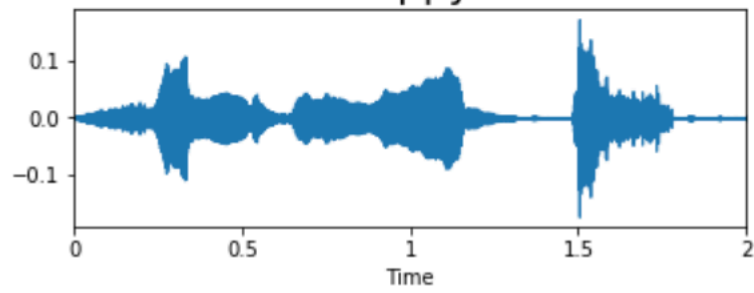
angry



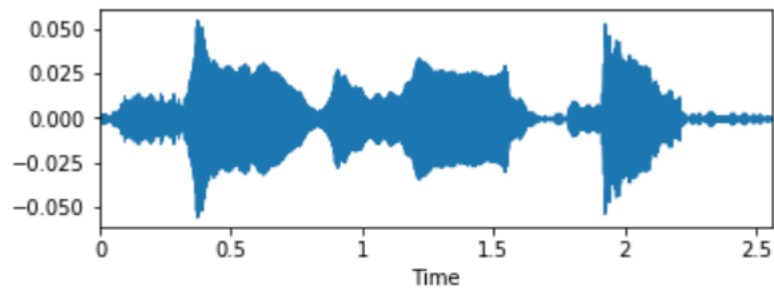
fear



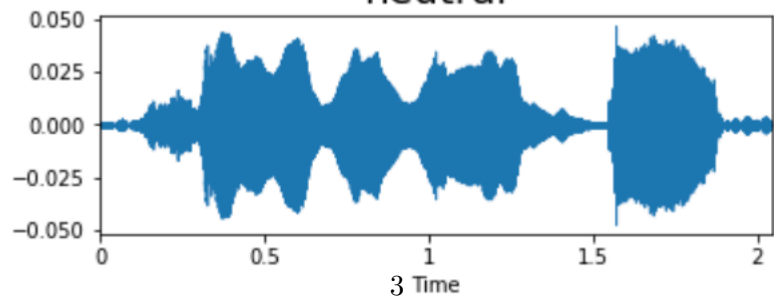
happy



sad



neutral



## 2.2 Pre-processing and feature engineering

In comparison to working with tabular data that might be used in machine learning directly, features should be extracted from sound files. As a preparatory step for feature engineering and training, metadata about audio files was created. Information for the metadata are extracted from file names that have a following pattern: talker\_word\_emotion.wav, e.g. OAF\_back\_angry.wav. Attributes currently used in the following steps are file name and class label (emotion).

Metadata examples:

filename	file_id	speaker	word	label
OAF_back_angry.wav	1	OAF	back	angry
OAF_back_fear.wav	3	OAF	back	fear
OAF_back_happy.wav	4	OAF	back	happy
OAF_back_neutral.wav	5	OAF	back	neutral
OAF_back_sad.wav	7	OAF	back	sad

Mel-Frequency Cepstral Coefficients (MFCC) is a well established feature for audio files that takes advantage of multiple ideas in sound preprocessing (overlapping windows, fast Fourier transforms, etc).<sup>3</sup> Despite being invented in the 70s, it is still often named state-of-the-art in articles on audio processing.[<sup>10</sup>]

The number of extracted coefficients in MFCC describe different number of sound aspects and various numbers were evaluated during the project. The modelling section below presents the influence of the number of extracted coefficients on the model performance.

## 2.3 Modelling

### 2.3.1 Benchmark

A no-model (random) prediction for a classification task could serve as a benchmark for all modelling approaches. With five categories (sad, happy, fear, angry, neutral) and a balanced data set, a baseline accuracy would be appx. 20%.

### 2.3.2 Algorithms and Techniques

Traditional methods of audio signal processing, e.g. Gaussian mixture models, hidden Markov models, have been consistently outperformed by neural network in case of sufficient data availability [<sup>10</sup>]. This motivated usage of deep learning for the multi-class classification in this project.

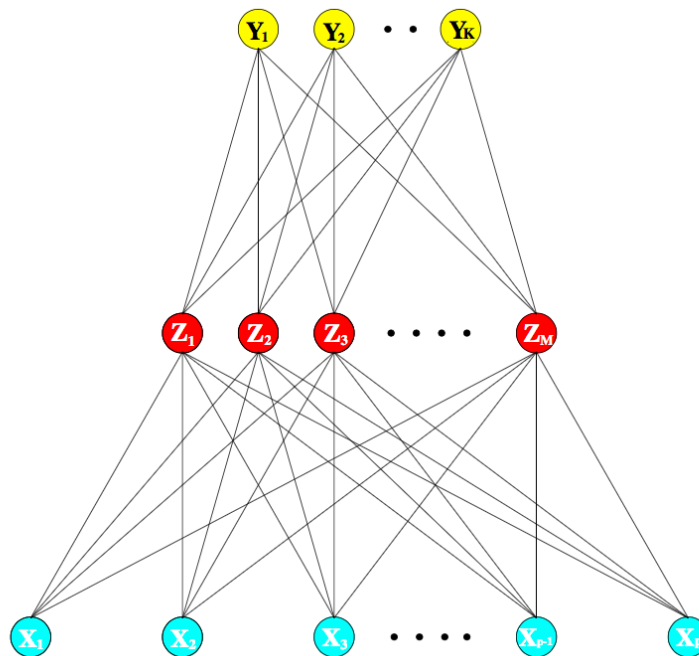
Neural network is an umbrella term for a large group of models. In essence, it is a non-linear statistical model that uses linear combinations of inputs as derived features. A network diagram

---

<sup>3</sup>Mendels, Gideon. How to apply machine learning and deep learning methods to audio analysis <https://towardsdatascience.com/how-to-apply-machine-learning-and-deep-learning-methods-to-audio-analysis-615e286fcbbc>

below gives a schematic representation of a general case of a single hidden layer back propagation network. The upper part ( $Y$ s) represents multiple classes (e.g. five emotions in this project), the lower part ( $X$ s) are inputs (e.g. MFCCs in this project) and the middle part are derived features from linear combinations of the inputs. This linear combinations are transformed by an activation function (ReLU in this project).<sup>4</sup> A number of hidden layers (one in the diagram below) and a number of units in the hidden layers are network parameters that allow to learn increasingly complex patterns in the data. At the same time, one should be especially aware of overfitting with a more complex network.

A neural network is fitted to the data by estimating the parameters (also known weights) using a measure of with an appropriate metric (e.g. multiclass cross-entropy in this project). This measure is minimized by applying a gradient decent (step of back-propagation). A learning rate (step size in response updated value of the loss function) is another hyperparameter of a neutral network.<sup>5</sup>



My guiding principle: a parsimonious model is preferred, however, it should be able to learn the complexities of the input well enough. Among a large number of neural networks types (Convolutional Neural Networks, Long short-term memory, etc.) a simpler approach close to the discussed in the diagram above was applied in this project. Various architectures of Multilayer Perceptron (MLP) with different number of MFCCs (column one in the table below) were compared to the no-model benchmark performance in terms of accuracy.

TESS data for five emotions was split to train (2/3) and test (1/3) with stratification. In addition, I recorded a few files myself, imitating different emotions. These were used for plausibility check after the final model was selected, in order to avoid influence of some peculiarities of the TESS dataset.

---

<sup>4</sup>Trevor Hastie, Robert Tibshirani, Jerome Friedman (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Pp. 389-397.

<sup>5</sup>Trevor Hastie, Robert Tibshirani, Jerome Friedman (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Pp. 389-397.

This project utilized the *keras* framework with the *TensorFlow* backend for its simplicity in use combined with robustness and integration with later AWS SageMaker model deployment. Performance above 85-90% could be considered sufficient.

Notes on models with their respective architectures:

## Model 1

A ridiculously simple model that aims at verifying that there is nothing strange going on with the train and test data. It includes an input layer and a single fully-connected layer with two neurons and a ReLU activation function and an output layer with a softmax activation function.

Layer (type)	Output Shape	Param #
dense_58 (Dense)	(None, 2)	102
dense_59 (Dense)	(None, 5)	15
Total params: 117		
Trainable params: 117		
Non-trainable params: 0		

## Model 2

Type of the activation functions as in model 1. The number of neurons in the hidden layer equals to the number of MFCCs (13, 20, 30, 40, or 50 respectively). A drop out layer is added to prevent overfitting. The architecture below depicts the version with 50 MFCCs.

Layer (type)	Output Shape	Param #
dense_60 (Dense)	(None, 50)	2550
dropout_29 (Dropout)	(None, 50)	0
dense_61 (Dense)	(None, 5)	255
Total params: 2,805		
Trainable params: 2,805		
Non-trainable params: 0		

### Model 3

Increased number of neurons in a hidden layer (128) compared to in model 2. Dropout layer.

Layer (type)	Output Shape	Param #
dense_62 (Dense)	(None, 128)	6528
dropout_30 (Dropout)	(None, 128)	0
dense_63 (Dense)	(None, 5)	645
Total params: 7,173		
Trainable params: 7,173		
Non-trainable params: 0		

### Model 4

Even higher number of neurons in a hidden layer (256). Dropout layer.

Layer (type)	Output Shape	Param #
dense_64 (Dense)	(None, 256)	13056
dropout_31 (Dropout)	(None, 256)	0
dense_65 (Dense)	(None, 5)	1285
Total params: 14,341		
Trainable params: 14,341		
Non-trainable params: 0		

### Model 5

Additional hidden with the same number of neurons than the first one (256). Dropout layer after each hidden layer.

Layer (type)	Output Shape	Param #
dense_66 (Dense)	(None, 256)	13056
dropout_32 (Dropout)	(None, 256)	0
dense_67 (Dense)	(None, 256)	65792
dropout_33 (Dropout)	(None, 256)	0
dense_68 (Dense)	(None, 5)	1285
Total params: 80,133		
Trainable params: 80,133		
Non-trainable params: 0		

The learning process was configured in the same way for all models: Adam optimizer, categorical cross entropy loss function and accuracy as an evaluation metric.

### 2.3.3 Results

Overview of the accuracy on the test dataset:

MFCCs	Benchmark (theory)	Model 1	Model 2	Model 3	Model 4	Model 5
13	0.2	0.2	0.28	0.73	0.81	0.44
20	0.2	0.2	0.39	0.88	0.94	0.32
30	0.2	0.2	0.65	0.96	0.95	0.62
40	0.2	0.24	0.91	0.98	0.97	0.80
50	0.2	0.2	0.94	0.97	0.99	0.95

Main observations from the experiments with combinations of feature engineering and modelling:

- As intended, Model 1 can be interpreted as a practical benchmark. With the accuracy of 20% it performs exactly as a no-model benchmark.
- In general, performance of Models 2 to 5 increases with larger number of MFCCs. As number of extracted coefficients rises, there is a danger that the amount of data would become insufficient for their estimation. Which in term would deteriorate generalizability of the model.
- There should be a balance between number of inputs and network complexity. For example, Model 5 performs poorly with low number of MFCCs. Overall, as more parsimonious models are preferred, considering no advantages in terms of performance by Model 5, it is not selected as a final model.
- There seem to be no large performance gains between Model 3 and 4, especially with higher MFCCs, therefore Model 3 is preferred.
- Model 2 performs well starting from quite high number of MFCCs, while Model 3 does so on much lower number of dimensions. This gives a preference to Model 3 with 30 MFCCs.
- Model 3 with 30 MFCCs was selected as the final model for deployment for the sum of reasons presented above. According to the confusion matrix below (actual values as rows and predicted as columns), the model classified some of the neutral expressions as sad (quite plausible and could be misclassified by a human), and predicted some happy and fearful expressions as angry.

col_0	sad	happy	angry	fear	neutral
row_0					
sad	132	0	0	0	0
happy	0	123	8	0	1
angry	0	4	124	3	1
fear	0	0	3	129	0
neutral	13	0	0	0	119

The validation of the selected model on the self recorded files demonstrated correct prediction for

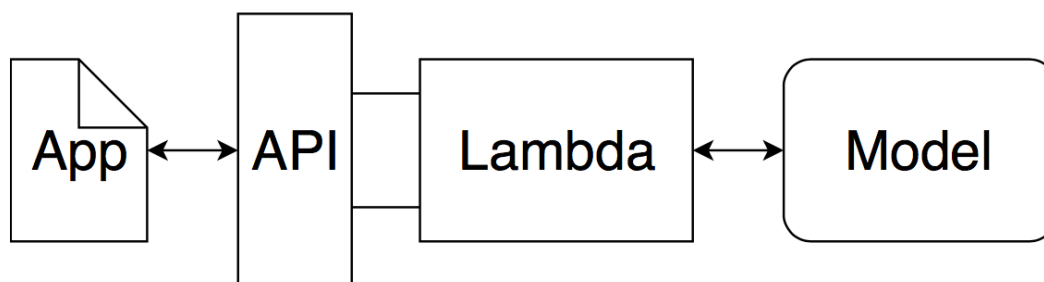


all three files. This is not representative in terms of numbers, but was used as a plausibility check that the model works on an audio file not belonging to the test and train sets. This last step in development builds a bridge to model deployment and serving in the next section.

### 3 Deployment and Serving

The productionizing of the solution includes model deployment and making prediction.

As described in the previous section, a keras model was trained. This project included a deployment of a keras model on AWS SageMaker and serving predictions via an app. The general overall serving architecture <sup>6</sup> is presented below:



Subsection below describe different aspects of achieving the target architecture: model deployment, making predictions, Lambda function, API gateway and web app.

An AWS blog post about deployment of a keras model on SageMaker <sup>7</sup> gave me a good general idea how to proceed. First, I changed the training approach to save model in a format enabling deployment on SageMaker. The blog post described a workaround to deploy a model, by using `sagemaker.tensorflow.model.TensorFlowModel` and simulating its creation with an empty `train.py` file. As I discovered from the Sagemaker documentation, there is now a better way to deploy directly from model artifacts with `sagemaker.tensorflow.serving.Model` <sup>8</sup>. Importantly, the specification of the `framework_version` parameter aligns the TensorFlow version of the trained model with the version of the serving model. Otherwise, an error was thrown. In my case, inclusion of the `framework_version='2.0.0'` was necessary, because the model was trained with TensorFlow 2.0.

An end user is interested in the emotion label. e.g. *happy*. My approach during the prototype stage was to call `.predict_classes()` on the model to get a class represented by an integer and then transform it back to string labels with `.inverse_transform()` of the created `LabelEncoder()`.

A model deployed in SageMaker has no method `.predict_classes()`, but only a `.predict()` method that outputs probabilities for all classes. This method corresponds to the `.predict()` on a local

---

<sup>6</sup>Udacity github repository <https://github.com/udacity/sagemaker-deployment/blob/master/Project/Web%20App%20Diagram.svg>

<sup>7</sup>Priya Ponnappalli (2019). Deploy trained Keras or TensorFlow models using Amazon SageMaker <https://aws.amazon.com/blogs/machine-learning/deploy-trained-keras-or-tensorflow-models-using-amazon-sagemaker/>

<sup>8</sup>Sagemaker Documentation. Deploying directly from model artifacts [https://sagemaker.readthedocs.io/en/stable/using\\_tf.html#deploy-directly-from-model-artifacts](https://sagemaker.readthedocs.io/en/stable/using_tf.html#deploy-directly-from-model-artifacts)

model. A class with a maximum probability value is predicted. For example, 0.53587806 is the maximum among (0.01964708, 0.03498399, 0.53587806, 0.01071843, 0.39877242) and the class 2 is predicted. With the help of a mapping dictionary, the class to is transformed to *happy*.

This logic developed for transforming probability prediction into labels was applied in the **Lambda** function that receives MFCCs from the web app, invokes the prediction endpoint, receives the probability prediction for all classes from the endpoint and transforms it to the emotion class prediction.

A public **API** to access the deployed model was created using Amazon API Gateway. It that triggers the Lambda function described above.

The predictions can be served to users via a **web app**. The *Submit* button from the first screenshot below makes a POST request to the API assessing the deployed model. An example of a prediction can be seen on the second screenshot below.

## Do you sound sad, happy, fearful, angry or neutral?

Enter the 40 mel-frequency cepstral coefficients (MFCC) of your audio file separated by a comma followed by a space (', ') below and click submit to find out...

Review:

Submit

## Do you sound sad, happy, fearful, angry or neutral?

Enter the 40 mel-frequency cepstral coefficients (MFCC) of your audio file separated by a comma followed by a space (', ') below and click submit to find out...

Review:

-550.37286, 48.238766, 18.442627, 6.504992, 0.82730311, -6.1545024, -24.476097, -7.476028, -11.998719, -9.4280348, -7.5956612, -5.5952277, -4.4645872, -6.2611837, -3.2972498, -8.0563803, -4.390583, -2.141741, -6.5392509, -6.4882522, -6.0786166, -8.1123333, -5.5041442, 0.046315003, -1.7572067, 0.89478511, -0.12644883, 2.5937853, -1.1436307, 3.4961894, 2.2928705, 3.7813089, 4.3201895, 1.7073919, 1.4069341, -0.13996479, -0.33032024, 1.0882645, -0.58361721, 1.6979592

Submit

You sounded happy!

## 4 Further work

Improvements could be done in various aspects of the project: data, feature engineering, modelling, and web app.

- **Data:** Extend list of emotions from the used TESS dataset.
- **Data:** Add data from another datasets containing emotion labels, e.g. consider RAVDESS Emotional song audio<sup>9</sup>.
- **Feature engineering:** Experiment with other features extracted from sound, an overview could be found in the Audio Features chapter<sup>10</sup>.

<sup>9</sup>Zenodo. RAVDESS Emotional song audio <https://www.kaggle.com/uwrfkaggler/ravdess-emotional-song-audio>

<sup>10</sup>Theodoros Giannakopoulos and Aggelos Pikrakis (2014). Introduction to Audio Analysis

- **Modelling:** Further experiments with MLPs, e.g. with optimizers or activation functions.
- **Modelling:** Experiment with other deep learning architectures, e.g. ones presented in a recent article <sup>11</sup> that reviews state-of-the-art deep learning technics for audio processing.
- **Web app:** improve the user interface by providing a possibility to record or upload an audio snippet for emotion detection.

---

<sup>11</sup>Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-yiin Chang, Tara Sainath (2019). Deep Learning for Audio Signal Processing. <https://arxiv.org/abs/1905.00078>