Voice emotion classification

Using neural networks

Saša Lalić

Computation and control engineering

Faculty of technical sciences

University of Novi Sad

[sasalali@gmail.com](mailto:sasalali@gmail.com)

***Abstract*-detecting emotions in voice files could have many applications such as prioritising or selecting data for processing, such as in customer support. We tested several ways of preprocessing input data, and compared the results of several neural network architectures commonly used in sound processing.**

***Keywords-neural network, emotion, classification, voice, rnn, cnn***

# Introduction

In order to explore potential application of neural networks in voice emotion recognition we have limited our work to techniques, and network models commonly used for audio processing. We took the training data from the IEMOCAP dataset, and applied several different preprocessing techniques. These datasets were then used to train and test different network models, using different optimization algorithms, with some minor changes to models in an attempt to find the best model. We then compared and discussed the results.

# Tools and technologies used

## Keras

[3] Keras is an open source neural network library that simplifies experimentation with neural networks. It contains commonly used building blocks such as layers, objectives, activation functions and optimizers as well as tools for working with images and text data. It provides a common interface that can run neural networks using Keras, Tensorflow and MXnet.

## Tensorflow

[4] Tensorflow is an open source symbolic math library used for dataflow programming and machine learning applications such as neural networks. It provides APIs for python, C++, Haskell, Java, Go and Rust. We will be using Keras to interface with Tensorflow in order to avoid dealing with the lower level of abstraction that comes with Tensorflow.

## CUDA

[2] The NVIDIA CUDA deep neural network library is a GPU accelerated library of primitives for deep neural networks. It provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers. CUDA accelerates widely used deep learning frameworks, including Caffe2, MATLAB, Microsoft Cognitive Toolkit, TensorFlow and Theano.

## Librosa

[1]Librosa is a python package for music and audio analysis. It provides implementations of a high variety of commonly used music information retrieval functions. We used it to handle converting audio files into spectrograms, and to assist with expanding the dataset by modifying existing data.

# Training data

In order to train the neural networks, we need a large amount of sample data representing emotions to allow the networks to learn how to recognize them. IEMOCAP presented a suitable set of data labeled by emotions.

## IEMOCAP

[5] The Interactive Emotional Dyadic Motion Capture database is an acted multispeaker database collected at the University of Southern California. It contains about 12 hours of audiovisual data, video, speech, motion capture of facial expressions and text transcriptions. It consists of sessions where actors perform scripted scenarios or improvisations meant to elicit emotional responses. It is annotated by multiple annotators into categorical labels such as anger and happiness, as well as into dimensional values such as valence. We have made use of the speech data, as well as the categorical labels to create our dataset.

## Preparing the data

The IEMOCAP dataset contains audio files that represent sentences from their scenarios. Each sentence of audio data has been categorized by 4 annotators to fit one of 8 potential categories of emotions, anger, excitement, frustration, happiness, neutral, sad, surrender, as well as unknown. We create our training data we assigned each sentence a single emotional category based on averaging the categories given by the annotators, discarding any data that couldn’t be given a specific category. Due to the surrender category containing about 200 values, compared to 2 to 4 thousand values for the other categories we created a separate dataset without the surrender category to test potential improvements to the neural networks score. Once split according to categories we transformed the audio files using librosa into numpy matrices representing mel-scaled spectrograms in order to take advantage of advanced image processing neural network models. The number of mels for each spectrogram was set to 96. Each of these values is then cut into segments that represent about 3 seconds of audio, with any segment smaller than 3 seconds being padded with zeros. This is necessary due to the neural networks requiring a specific input size.

## Expanding the dataset

In order to test if the size of the dataset was limiting the neural network scores another dataset was created that expanded on the first one by augmenting the existing data to create additional values. The mels for these spectrograms were set to 128. Each value was used to create a new value by changing its speed, pitch, dynamic range, by adding noise and by shifting it in time randomly.

# Neural network models

## CNN

Convolutional neural networks are networks that are made for image analysis, their architecture vaguely based on biological image processing neural networks. Our base model consists of an input convolutional layer, followed by 3 convolutional layers after each of which is a 2d pooling layer and a dropout layer. Finally there is a dropout layer at 25% followed by a softmax activation layer that performs the final categorization. Each of these convolution layers has 32 filters and 3 with 3 kernel size. The pool size for the pooling layer is 2 with 2. This network was usually trained with RMSprop optimizer.

## RNN

Recurrent neural networks have an internal state that allows them to process sequences of input, this makes them applicable for speech recognition. Our base model consists out of a LSTM(Long short-term memory) layer with 256 units a dropout layer, another LSTM layer with 128 units, a dropout layer and a softmax activation layer. This network used Adam as the optimizer.

## VGG16

[6] VGG-16 is a pretrained image recognition model, which we modified by replacing its final layer with our own softmax activation layer. The input data had to be fitted to the expected size by padding the input matrices with zeros.

# Results

The input data was split into a training and validation set using a 80:20 split. The results presented in the table are the percentage of correctly classified values from the validation set. Each network was trained for at most 50 epochs, or less if training was stopped to prevent overtraining. Several changes to CNN architecture were tested to see how they influence results. Apart from the changes listed in the tables we tried replacing ReLU activation functions with ELU functions, as well as leakyReLUs. There was no notable difference in network output with leakyReLUs although the ELU functions seem to reduce the networks accuracy. The loss function used was categorical crossentropy due to its suitability for categorizing data into multiple categories.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | base dataset | expanded dataset | base dataset without surrender | expanded dataset without surrender |
| CNN | 38.8% | 32.6% | 41.1% | 36% |
| RNN | 32.7% | 33.3% | 32.5% | 33.36% |
| VGG16 | 19.2% |  | 24% |  |
| CNN 16 filters, dropout to 50% | 32.8% | 33.1% | 36.5% | 28% |
| CNN,64 filters, dropout 50% | 36.9% | 34% | 39% | 32.8% |
| CNN with 6 layers | 37.7% | 35.5% | 35.4% | 33.3% |
| CNN with 64 filters | 35.8% | 25.2% | 36.5% | 31.3% |
| CNN with 16 filters | 33.3% | 35% | 34.2% | 35.6% |
| CNN, 16 filters, 10 layers, no pooling | 29.4% | 34.9% | 38% | 34.1% |
| CNN 16 filters, no pooling | 31.8% | 33.6% | 35.6% | 30.6% |
| CNN, 16 filters, dropout 50%, kernel size 2,2 | 35.7% | 38.2% | 33.5% | 30.2% |
| CNN, 20 filters, kernel size 2,2, dropout 50% | 38.2% | 32.4% | 35.8% | 30.1% |
| CNN, dropout 50%, SGD optimizer | 30.7% | 31.9% | 30.4% | 28.3% |

The results given in the table mostly vary from 30 to 40%, with a few particularly poorly suited architectures going below that. Training the pretrained model was impossible due to hardware restrictions, namely the input data for vgg-16 is too large to allow the expanded set to fit into memory. But based on training on the base dataset we can see that vgg-16 is unsuited to classifying sound, most likely due to it being trained to recognize classic images such as birds or vehicles, and not being flexible enough to extract features from a spectrogram.

RNN on the other hand appears to be better suited to classification, achieving over 30% accuracy.

The best results are achieved by some of the attempted CNN models, the base model achieving the best results given a suitable training dataset. Removing the class with too few example values from the dataset can improve accuracy on some architectures, such as when there are only 16 filters. Decreasing pooling size increases training time with no significant impact on accuracy. Kernel size doesn’t significantly impact accuracy either. Number of layers is limited due to input size not allowing too many reductions in size, and reducing pooling increases overfitting issues. More layers influences accuracy negatively. Higher dropout seems to mildly increase accuracy but only in combination with other changes that reduce it, such as lower or greater filter number. The best optimizer is RMSprop, achieving much better results than other optimizers like SGD.

# Discussion

We have explored the suitability of several neural network architectures for classifying emotions. The main issues are acquiring a good training set, due to a lack of datasets with enough examples. Next is representing the data in a way that can be input into a neural network classifier, due to varying length of audio data, and a need for strict input size for some network types. Finding an appropriate architecture for the neural network is only a matter of trial and error, due to any large changes to commonly accepted architectures leading to reduction in accuracy.

Any further research could focus on a better representation of audio data, such as extracting specific features, or removing excess data. Exploring classification in different languages, as well as attempting to create a classifier that worked across languages could also bring new insights.

# References

[1]McFee, Brian, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. "librosa: Audio and music signal analysis in python." In Proceedings of the 14th python in science conference, pp. 18-25. 2015.

[2] NVIDIA cuDNN, h[ttps://developer.nvidia.com/cudnn](https://developer.nvidia.com/cudnn)

[3] [Keras backends](https://keras.io/backend/) keras.io. Retrieved 2018-04-12.

[4] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo,

Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis,

Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow,

Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia,

Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster,

Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens,

Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker,

Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas,

Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke,

Yuan Yu, and Xiaoqiang Zheng.

TensorFlow: Large-scale machine learning on heterogeneous systems,

2015. Software available from tensorflow.org.

[5] C. Busso, M. Bulut, C.C. Lee, A. Kazemzadeh, E. Mower, S. Kim, J.N. Chang, S. Lee, and S.S. Narayanan, "IEMOCAP: Interactive emotional dyadic motion capture database," Journal of Language Resources and Evaluation, vol. 42, no. 4, pp. 335-359, December 2008.

[6] VGG -16 <https://keras.io/applications/#vgg16>