



UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering,
Mathematics & Computer Science

Aggression Based Audio Ranking

Annotation and Automatic Ranking

Daan H. Wiltenburg
M.Sc. Thesis
August 2019

Supervisors:

dr. M. Poel
dr. K.P. Truong
dr. ir. J. van Dorp Schuitman

Human Media Interaction Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

Summary

In many places, public and private, aggression can be a big problem. Therefore, surveillance is essential. Sound Intelligence builds software especially for this purpose. Currently an aggression classifier is used as surveillance assistant. In this study, techniques from the field of information retrieval are applied in order to improve the performance of this classifier. More specifically, aggression based audio ranking is investigated.

To automatically rank audio files, a ground truth is needed. Therefore this research is twofold. The first part of this research focuses on annotation of the data set. Different methods of rank annotations are compared on transitivity and efficiency. The annotation methods are based on pairwise and list-wise comparisons. The methods are compared on time efficiency and transitivity. 50 subjects participated in this experiment. First a ground truth is established by pairwise comparing every sample in a subset of the data. This subset is used to evaluate different, more efficient methods. Furthermore, a simulation is done to predict the time needed to annotate a bigger data set.

The results show that a pairwise comparison method based on binary insertion sort is the most efficient way to annotate audio samples into a ranking. To annotate a set of 300 audio samples, this method would take 8 hours whereas the second best method would take 21 hours. Furthermore this method yields the most transitive ranking, with an in-transitivity value of 0.146 which is better than the baseline(0.15) and the list-wise methods(0.157 and 0.169). The pairwise method is used to annotate 300 audio samples into a ranking. The rankings of 4 annotators are used to assemble a data set that is used for supervised machine learning.

The second problem addressed in this research is in the field of machine learning and learning-to-rank. Two type of loss functions are compared. The first loss function is Mean Squared Error, which uses a continuous target between 0 and 1 to train a network making this a regression approach. The second loss function is the log-likelihood function, which takes an ordered list as target making this a list-wise approach. For both approaches two network architectures are developed and optimized: A fully connected neural network and a convolutional neural network. For optimization hyperopt¹ is used. The combination of these architectures and loss functions yields 4 models.

The models are evaluated based on the final ranking they yield. The measures used are Kendalls Rank Correlation Coefficient, Spearmans Rank Correlation Coefficient and Mean

¹Hyperopt is a python library that can be used to perform hyper-parameter optimization.

Squared Error. The best model achieved an KRCC of 0.6049, an SRCC of 0.8228 and an MSE of 0.0364. Furthermore, the results are compared to the performance of the annotators. This comparison shows that the models yield rankings that correlate moderate to strong with the rankings of annotators.

The research shows that audio samples can be ordered automatically based on aggression. Furthermore it is shown that machine learning can be applied to automatically rank audio samples based on aggression.

Contents

Summary	iii
1 Introduction	1
1.1 Sound Intelligence	1
1.2 Goal of the Research	1
1.3 Reader's Guide	2
2 Background	3
2.1 Background in Data Annotation for Audio	3
2.1.1 Individual Annotation	4
2.1.2 Paired Comparison	5
2.1.3 Binary Insertion Sort	5
2.2 Background in Audio Processing	6
2.3 Background in Machine Learning	6
2.3.1 Support Vector Machines	7
2.3.2 Neural Networks	7
2.3.3 Convolutional Networks	7
2.3.4 Recurrent Neural Networks	8
2.4 Audio Processing Using Machine Learning	8
2.4.1 Deep Learning Audio Event Detection	8
2.4.2 Convolutional Neural Networks and Audio	9
2.4.3 Weakly labeled data	10
2.4.4 Varying Length Audio Input	10
2.5 Ranking and Machine Learning	11
2.5.1 Pointwise	11
2.5.2 Pairwise	11
2.5.3 List-wise	12
2.6 Loss Functions	12
2.6.1 Loss Functions for Regression	12
2.6.2 Loss Functions for list-wise Learning-to-rank	13
2.7 Hyper Parameter Optimization	14
2.8 Background Ranking Statistics	14
2.8.1 Similarity Metrics	14

2.8.2	Rank Evaluation	17
3	State of the Art	19
3.1	Aggression detection in audio	19
3.2	State of the Art in Audio Annotation	19
3.2.1	Hybrid Comparison	20
3.3	State of the Art in Automatic Audio Ranking	21
3.4	Conclusion of State of the Art	22
4	Research Questions	23
4.1	Research Goals	23
5	Research Methodology	25
5.1	How can 3000 audio files be annotated in a ranking in an efficient manner?	25
5.1.1	Baseline	25
5.1.2	Binary Search	25
5.1.3	Hybrid Comparison Insertion	25
5.2	Automatic Ranking of Aggression	27
5.2.1	Data	27
5.2.2	Data Analysis	28
5.2.3	Audio Ranking using Machine Learning	29
5.2.4	Performance	30
5.2.5	Hyper Parameter Optimization	31
6	Results: Efficient Rank Annotation	35
6.1	RQ1: How can 3000 files be efficiently annotated?	35
6.1.1	Transitivity	36
6.1.2	Time Efficiency	38
6.1.3	Conclusion Exploration of Annotation Methods	38
6.2	Expected time for 3000 samples	39
6.2.1	Simulate annotation	40
6.2.2	Conclusion	42
7	Results: Data analysis	43
7.1	Individual <i>KRCC</i> results	43
7.2	Individual <i>SRCC</i> results	43
7.3	In-transitivity versus the different normalized rankings	44
7.4	<i>KRCC</i> and <i>SRCC</i> against ground truth	44
7.5	Qualitative Feedback From Annotators	44
7.6	Conclusion	45

8 Results: Machine Learning	47
8.1 Hyper parameter optimization	47
8.1.1 Regression	47
8.1.2 List-wise	48
8.2 Results five fold cross validation	48
8.2.1 Regression	48
8.2.2 List-wise	49
8.2.3 Results against test set	51
8.3 Results sensitivity test	52
8.4 Remove difficult to rank samples	52
9 Conclusion	55
9.1 RQ1: How can 3000 files be efficiently annotated?	55
9.1.1 SQ1.1: What is the level of transitivity in aggression annotation in audio samples?	55
9.1.2 How can audio samples be annotated into a ranking based on aggression?	55
9.1.3 How efficient are the different types of audio annotation when ranking based on aggression?	56
9.2 How can audio samples automatically be ranked based on aggression?	57
9.2.1 How can regression be applied to rank audio files based on aggression?	57
9.2.2 How can list-wise learning-to-rank be applied to rank audio files based on aggression?	57
9.2.3 List-wise versus Regression	57
9.2.4 Sensitivity test	58
9.2.5 Easy data	58
10 Discussion	59
10.1 Results Compared to Kooij et al.	59
10.1.1 Data annotation very time consuming	59
10.2 <i>SRCC</i> vs <i>KRCC</i>	60
10.3 Automatic Ranking of Audio Based on Aggression	60
10.4 Value based treshold	61
10.5 Future Research	61
10.5.1 Different Machine Learning Techniques	61
10.5.2 Data Source	61
10.5.3 Exclude Irrelevant Samples	61
11 Acknowledgement	63
References	65
Appendices	

A Statistical Formulas	71
A.1 Z-scores	71
A.2 Paired Students T-Test	71
B Pairwise Comparison Experiment Tool	73
C Binary Comparison Experiment Tool	75
D List-wise Comparison Experiment Tool	77
E List-wise Comparison Experiment Tool	79
F Annotation Tool	81
G Simulation of Annotation Process	83

Introduction

Although Speech Emotion Recognition(SER) already in the fifties, lately there is a growing interest in this research field. In 2017 more than 150 papers have been published in SER [1]. This review summarizes different classification techniques used in emotion recognition. All techniques described in this paper focus on discreet classification into different emotion labels. Nassif et al. [2] describe supervised machine learning in SER application. They define classification and regression as the main categories of machine learning. Another field in supervised machine learning is learning-to-rank, in which a model predicts a ranking rather than a label or continuous value, is relatively unexplored in SER.

1.1 Sound Intelligence

Sound Intelligence (SI) is a company that uses supervised machine learning techniques for event detection in audio. One of the events that the software of SI can detect is aggression. Currently the software works as a classifier, where some event is either aggression or it is not. The user of the system can modify the sensitivity of the classifier, which determines the trade off between false positives and false negatives. SI wants to extend this software to a more advanced sensitivity system, where this functionality is not a trade of between two wrongs. The new functionality should allow the user to set a threshold from which aggressive events should be reported. This research is aimed to find ways of automatically ranking audio files based on aggression. This ranking will make it possible to set such a threshold, which is a more sophisticated method of control than the current sensitivity control and should lead to fewer false or missed reports.

1.2 Goal of the Research

Supervised machine learning is a specific type of machine learning where the correct output given an input is known [3]. To apply supervised machine learning techniques a labeled data set is needed. The labeling of this data is referred to as data annotation. A set of 2427 unlabeled audio samples is available at SI for this research.

The goals of this research are two-fold: First, the different possibilities of data annotation will be investigated in order to determine an efficient way to rank the data set. This data set will be used as a training-, validation and test set for different supervised machine learning techniques.

Secondly, different supervised machine learning techniques are investigated to determine how to automatically rank audio based on aggression.

1.3 Reader's Guide

For readability this report is structured according to these two aspects of the research. First, a background is given on data annotation, followed by a background on (supervised) machine learning. Then, the state of the art in data annotation is presented, followed by the state of the art in audio ranking using supervised machine learning. After that, the research goals for data annotation and the research goals for supervised machine learning are given. Following, the research methodology for respectively data annotation and supervised machine learning are described. After the methodology, the results of the data annotation are presented followed by the results of supervised machine learning experiment. Lastly, a conclusion and discussion of the research project are given.

Background

2.1 Background in Data Annotation for Audio

Susini et al. [4] describe different measurements that are used in psychoacoustics, a field in which ..research aims to establish quantitative relationships between the physical properties of a sound [...] and the perceived properties of the sound [4]. First the classical evaluations methods are described, divided into indirect and direct methods:

- Ratio scaling: In this scenario the sensation related to a sound has to be expressed on a numerical judgment scale.
- Cross Modal Matching: According to the dictionary of psychology this is A scaling method [...] in which a [participant] makes the apparent intensity of stimuli across two sensory modalities, as when an [participant] adjusts the brightness of a light to indicate the loudness of [an audio sample]. [5]

Previous methods are usable for stationary sounds, but everyday sounds are not stationary, neither is aggression. Non stationary sounds require continuous rating of sound. The paper describes five categories of continuous sound judgment:

- Method of continuous judgment using categories: Participants judge an attribute of a sound in real time into categories ranging from very loud to not loud at all, they change to another category when their evaluation of the sound changes.
- Audiovisual adjustment method: Participants adjust the length of a line proportional to the auditory sensation in real time.
- Continuous cross-modal matching: this is similar to cross modal matching of a stationary sound; only now the participant changes the evaluation modality along the length of the stimulus.
- Analog categorical scaling: Participants can slide a cursor continuously along five discrete categories [4]. As opposed to continuous judgment using categories, the results are now measured on an analog instead of a discrete scale.

- Semantic scale used in real time: This is used to track real-time emotional response to music or sound. In this set up a participant has to evaluate the related sensation on one or more emotion scales in real time when a sound sample is presented.

Next to classical evaluation methods, multi-dimensional and exploratory methods are described.

- Semantic differential (SD): The participant has to rate stimuli on a semantic scale with two opposed descriptors. Examples of these descriptors are good-bad or pure-rich. When using SD the participant has to evaluate one stimulus at a time.
- Dissimilarity judgments: Contrary to SD no descriptors are used; now two samples are compared with each other and rated on similarity scales. Euclidean distance is calculated to determine similarity. This leads to $N(N - 1)/2$ comparisons.
- Sorting tasks: listeners are required to sort a set of sounds and to group them into classes. [4]
- Free sorting tasks: no predefined (number of) classes. Various methods of free sorting and grouping of stimuli into different categories are explained. Free sorting is out of the scope of this research, for more information the reader should review [4].
- 2AFC: two alternatives, forced choice. Two stimuli are presented, the listener has to choose one of the two as the best option related to a certain scale (i.e. choose the most aggressive sample).

Following the paper of Susine et al. [4], some interesting categories of sound annotation methods can be defined. These techniques are tested in other papers as well.

2.1.1 Individual Annotation

This is the technique where one stimulus is judged at a time. Continuous sound judgment and SD are techniques described by Susini et al. [4] that fall into this category. A widely used measure for individual affect annotation is the valance (positive/negative) and arousal (high/low) two dimensional scale, proposed by James Russell [6] which can be used to describe every emotion. Because the annotator does not have to name a specific emotion, language ambiguity is omitted.

Individual evaluation is the most time efficient way of annotating data. Since there are no comparisons, the amount of annotations needed is equal to the amount of stimuli. however multiple studies [7]–[10] have shown that this evaluation leads to in-transitive¹ results.

¹(in)Transitivity is explained in chapter 2.8.1.

2.1.2 Paired Comparison

Susini et al. [4] describe Dissimilarity judgments and 2AFC, which is a form of paired comparison. Parizet, Guyader and Nosulenko [8] used pairwise comparison in their experiment to find the sound of a car door closing, that evokes the best quality. The comparison had to be answered by either choosing one sample, or both samples were perceived equal. 40 participants evaluated 12 sounds, resulting in 66 pairwise comparisons $(N(N - 1)/2)$. If x is preferred over y , then the preference score $p_{x,y} = 1$. If y is preferred over x , then the preference score $p_{x,y} = 0$. If x and y are indifferent, $p_{x,y} = 0.5$. The preference scores are averaged over the 40 participants. The correlation between the measured and the estimated probability was 0.94, indicating that this is a very accurate way of comparison. The fact that 66 comparisons are needed to evaluate 12 sounds is a major drawback.

Poirson et al. [9] compare the evaluation of 11 diesel engine sounds by experts and naive subjects, where experts are subjects trained in the evaluation of audio samples. The experiment included two panels: an expert panel, consisting of 10 trained subjects; a panel of naive subjects, consisting of 30 students. They found that the evaluations of experts and naive subjects are very similar when it comes to paired comparison. only 5 out of 30 naive subjects are discarded for giving in-transitive results, indicating that naive subjects can be suitable annotators for paired comparison tests. On the other hand they found that naive subjects find it difficult to evaluate individual sound samples on a numeric scale, leading to in-transitive evaluations.

As opposed to Parizet et al. [8], the participants of [9] did not have to compare the entire comparison matrix. To get computable results the subjects had to compare one full row of the matrix and 12 additional comparisons. This leads to $10 + 12 = 22$ comparisons, instead of $(11 * 10)/2 = 55$ comparisons.

As literature shows, comparisons are more discriminant than judgments. On the other hand this is very time consuming. Furthermore, in most of the above papers the author is working with up to 12 stimuli, where as sound Intelligence is looking for a solution to annotate thousands of files. It is not feasible to compare every stimulus with each other. Therefore two methods of inserting new data into a ranked list will be evaluated on efficiency, which are explained in 5.1.

2.1.3 Binary Insertion Sort

There is a variety of sorting algorithms used in computer science, Binary Insertion Sort is such a sorting algorithm that is widely used. In this method an unranked sample x will be compared to multiple samples in the ordered list. The comparison will start with comparing the sample y that is in the middle of the ordered list. If x has a higher value than y , x will be compared to another sample in the ordered list: the sample right in between y and the end of the ordered list. If x has a lower value than y , x will be compared to the sample right in between y and the beginning of the list. It will repeat this process until there are no more

samples to compare with, which will be the position where the new sample is added to the ordered list. This can be used to efficiently build a ranking: $\mathcal{O}(n \log n)$.

2.2 Background in Audio Processing

Before mathematical operations can be applied to audio, it has to be transformed to a digital format. When the data is transformed from an analog to a digital signal, the data can be processed digitally. Often this digital signal is transformed into a more usable format. "Signal analysis transforms a signal from one domain to another, for example from the time domain to the frequency domain. By transforming the signal, the intent is to emphasize information in the signal and cast it into a form that is easier to extract" [11]. In most machine learning applications the wave input files are transformed into the frequency domain because this is a representation of the audio input over which machine learning models can efficiently generalize. To transform the data from the time domain to the time-frequency domain the data has to be processed in small frames. To transform these frames into the frequency domain Fast Fourier Transform (FFT) and Discrete Fourier Transform (DFT) are widely used. These techniques are explained in [11].

A type of audio input that is often used in audio classification are Log-Mel-Spectrums. First the audio signal is transformed to the frequency domain. This is done for a short window of the audio signal using Fourier transformation. The energy spectrum is obtained by taking the this transformation squared. The energy obtained by this transformation are transformed to the Mel scale, which is divided into multiple filterbanks. The logarithms of the powers in these filterbanks form the Log-Mel-Spectrums(LMS) [12], [13].

In speech recognition Mel-Frequency Ceptrum Coefficients (MFCC) are widely used. The MFCC of a signal is obtained by taking the spectrum of the LMS. A spectrum of a spectrum is called a ceptrum [14].

El Ayadi et al. [15] describe different features for emotion detection in audio. They describe energy and pitch as main continuous features. They mention MFCC as widely used frequency feature along with Linear Predictor Ceptral Coefficients (LPCC). Lately there have been experiments on raw audio wave input [16].

2.3 Background in Machine Learning

Machine Learning aims to find patterns in data. To do so different techniques are used, which will be explained in this chapter. All of the models take representation of data as input together with a target value. The models transform the input so that it can predict the output. In this research the input is referred to as feature vector.

2.3.1 Support Vector Machines

A Support Vector Machine (SVM) algorithm is an algorithm that finds the hyper plane that separates data points of two classes, with the maximum margin between the two classes. SVM's are widely used, but since computational expenses become less of a problem new architectures are replacing SVM's [17].

2.3.2 Neural Networks

Kevin Gurney [18] published a book called *An Introduction to Neural Networks* in which he describes the basic concept of Neural Networks. This book gives a clear and understandable explanation to readers who are new to this field. A Neural Network (NN) is a combination of simple mathematical operations in order to learn patterns from a set of training data. An NN consists of multiple layers and these layers typically consists of multiple nodes. The nodes in different layers are connected by weights. Each node receives input from multiple other nodes in the previous layer. Within a node the inputs are summed. Whenever this sum exceeds a certain threshold the neuron will generate an output that can be used for classification. A typical NN will consist of multiple layers: An input layer, which is a representation of the data. One or more hidden layers in which operations on the input will take place and an output layer. This output layer can be one node in a binary classification problem, but can also consist of multiple nodes. For example in character recognition this layer might contain 26 nodes, one for each character of the alphabet. Furthermore an NN can be used in a regression task where the network does not output a classification but predicts a value for a certain input. If all the nodes in layer x are connected to all the nodes in layer $x + 1$ for all layers, the network is called a fully connected NN. If the network consists of more than one hidden layer, the network is called a Deep Neural Network (DNN).

2.3.3 Convolutional Networks

Sandro Skansi [19] gives a detailed explanation about Convolutional Neural Networks (CNNs) and their mathematical background in his book *An introduction to Deep Learning*. CNNs are a specific type of NNs that are widely used, especially in image processing, but in many other fields as well. A CNN is an NN which has one or more convolutional layers. A convolutional layer is similar to a fully connected layer as it also consists of nodes that are connected with the previous layer through weights, but now these nodes are not connected to all nodes in the previous layer simultaneously. The filter size of a convolutional layer is smaller than the previous layer and moves step wise over the previous layer in order to learn local features. The convolutional layer can apply multiple filters to the same filter window.]

2.3.4 Recurrent Neural Networks

Skansi [19] also explains the use of Recurrent Neural Networks (RNNs). An RNN is a type of NN that feeds the output of data point x back into the network when processing data point $x + 1$. This way the network has information about previous inputs, giving it some sort of memory. Next to that, this makes it possible to process data of different sizes without the need to re-size the input data. This is specifically useful in audio processing for two reasons: First of all when processing audio it can be expected to have samples of different sizes. Next to that silence is an important aspect of audio so padding audio samples with zeros, in order to get same sized data inputs, will change the meaning of these audio samples. This can lead to a drop in accuracy. Two commonly used types of recurrent layers are Long-Short Term Memory (LSTM) and Gated Recurrent Units (GRU).

Sak et al. [20] describes the use of LSTMs in a speech recognition task. An LSTM architecture consists of three gates. An input gate determines which parts of the input to this LSTM layer should be kept. A forget gate, which decides what data is not usable and can be omitted. The final gate is an output gate which determines what should be passed on. An LSTM has two outputs. A cell state which, will be passed on to the LSTM layer of the next data input. Next to the cell state it outputs a hidden state which is used for prediction.

Wu and King [21] apply LSTMs as well as GRUs in a speech processing task. They describe GRUs as similar to LSTM architectures only now the forget and input gates are combined into a reset gate, which reduces the amount of operations making GRUs computationally less expensive. In this reset gate, information is only forgotten if something new is inserted in it's place.

2.4 Audio Processing Using Machine Learning

2.4.1 Deep Learning Audio Event Detection

Lane et al. [22] describe the development of an audio event detection system intended to run on mobile devices: "Deep Ear: The first mobile audio sensing framework built from coupled DNNs that simultaneously perform common audio sensing tasks" [22]. The system consists of four DNNs, every network performs a specific task: Ambient audio sensing analysis; speaker identification; emotion detection; stress detection. The framework is optimized for mobile devices by taking into account battery cost and use in a variety of environments. One of the research questions discussed in this paper is: *Can deep learning assist audio sensing in coping with unconstrained environments?*

The system first uses an energy threshold to detect silence. If there is no detection of silence, the first DNN will be activated. This is the Ambient Scene Analysis, which can detect the following: Voicing, music, water, traffic. If voicing is detected the other three DNNs are activated. Stress Detection returns a binary state, Emotion Detection returns

anger, fear, neutral, sadness or happiness. Speaker Identification can discriminate between 23 speakers.

To train the system both labeled and unlabeled data is used. The unlabeled data is used as background noise for the labeled data as well as to initialize the DNNs. All DNNs consists of the same architecture: 3 hidden layers with 1024 nodes in each layer. ReLU is used in combination with dropout to optimize the networks.

For validation a benchmark experiment is done, in which Deep Ear is compared to four systems with shallow architectures as oppose to the deep architecture in Deep Ear. These four systems each addressed one specific task that relates to one of the DNNs in Deep Ear. The results show a gain in accuracy for all four aspects, ranging from 7.7% to 82.5% for Ambient Scene Analysis to Speaker Identification respectively.

Deep Ear outperforms systems for comparable tasks greatly when these are trained on clean data. When the same benchmark systems are trained on noisy data, their performance improves but are still outperformed by Deep Ear. The Ambient Scene Analysis achieves an accuracy of above 80%, Stress and Emotion Detection both approximately 80% and Speaker Identification approximately 50%. The paper proves that using deep learning techniques greatly outperform shallow systems and can better cope with data with noise on different levels, indicating that deep learning can improve audio sensing with unconstrained environments.

2.4.2 Convolutional Neural Networks and Audio

Piczak [23] developed a system using a Convolutional Neural Network (CNN) to classify environmental sounds from three publicly available data sets: ESC-50, ESC-10 and Urban-Sound8K. The performance of the CNN is compared to a baseline system. The baseline system is a Random Forest with MFCC as input.

The proposed system consists two 2 convolutional layers with max pooling, combined with two fully connected layers. The system uses segmented spectrograms with deltas as input. Furthermore the system makes use of dropout and uses ReLU as activation function.

ESC-50 consists of 2000 clips(five seconds each) with a total of 50 balanced classes. The baseline model used in this research scored 44% on this data, human participants achieved an accuracy of 81%, the best CNN proposed in this paper reached an accuracy of 64.5%. ESC-10 consists of 400 of these records and a total of 10 classes, the baseline system reached 73% accuracy, human annotators reached 96% accuracy and the proposed system reached approximately 80%. UrbanSound8k consists of 2732 clips of four seconds or shorter, with a total of 10 classes. The baseline system² achieved an accuracy of 73.7%, the proposed system an accuracy of 73.1%.

Choi et al. [24] describe the comparison of a Convolutional Recurrent Neural Network (CRNN) to three different CNNs. The networks are tested on a music data set to predict the

²Different baseline than for the previous two data sets.

top 50 tags in this set, including genres, moods, instruments and era's. Approximately 215 thousand clips are used. As a metric, Area Under Receiver Operating Characteristics Curve (AUC-ROC) is used, because the clips are multi-label tagged.

The CRNN uses 4 CNN layers, after that follow two RNN layers with Gated Recurrent Units (GRU). The results show that the CRNN outperforms the CNNs on all experimented amounts of parameters, but needs more training time to achieve this. The highest AUC-ROC achieved by the CRNN is 0.86.

2.4.3 Weakly labeled data

Kong et al. [25] propose a Joint Detection Classification Model (JCD), which finds the informative and uninformative parts of an audio sample and only uses the informative parts to classify the sample. They describe two types of audio annotation: Clip level annotation, where each clip as one or more labels; Event level annotation, where each clip has one or more labels and every label has an occurrence time. The proposed system is based on human analyses of sound, where the writer assumes that humans detect when to attend to a sound. Subsequently humans classify the sound.

The system takes as input data that is annotated at clip level. The JCD algorithm can output labels at event level, leading to an Equal Error Rate (EER) of 16.9% which is lower than the baseline(19%) without the need of event level annotation.

2.4.4 Varying Length Audio Input

Varying length audio samples can lead to problems in Machine Learning. Since silence is part of audio, padding audio samples to a specific length might not be the best solution. [26] describe the use of a convolutional layer on the input. For the use of acoustic scene classification and domestic audio tagging they use the short-time Fourier transform (STFT) of steps of 25ms as one dimension. The amount of steps depends on the length of the sample, which determines the other dimension. The convolutional layer can handle this varying length, as long as the layer is smaller or equal to the least amount of steps occurring in the data set. The pooling layer will be applied on a dynamic pooling window.

Phan et al. [27] describe another shallow CNN architecture which uses a convolutional layer directly on the input. The network consists of three layers, one convolutional layer, one one-max-pooling layer and a Softmax layer. By using convolutional filters directly on the input, inputs can be of different sizes as long as they are longer than the longest filter. The convolutional layer is followed by a one-max-pooling layer, which reduces each filter to the size of one regardless of the input length. The described architecture makes use of four filters, this leads to a vector of length four after the pooling layer.

To train, validate and test the system respectively 2000, 500 and 1500 samples are used coming from 10 balanced classes. Noise is added from the NOISEX-92 database. four types

of noise are selected. The results show a great improvement of performance compared to the state of the art DNNs (76.3% Relative Error Reduction). The best system achieved a mean accuracy of 98.6% over four different noise levels.

2.5 Ranking and Machine Learning

A field in which ranking algorithms are widely used is information retrieval (IR). Thread or result ranking of big data sets are central problems in this field. These techniques can be relevant for Aggression ranking as well.

Jian Jiao [28] describes a system that is used for the ranking of usefulness of online treads based on specific queries. Furthermore he states that the state of the art ranking algorithms can be divided into three categories:

- Pointwise
- Pairwise
- List-wise

these algorithms are very similar to the described annotation schemes for ranking purposes.

2.5.1 Pointwise

For every input a (relevance) score is calculated. The inputs are ordered based on these scores, yielding a ranked list. The ground truth is an ordinal score or numeric value. This output can be human annotated or automatically generated based on some mapping. Li et al. [29] propose such a system.

2.5.2 Pairwise

The input of this algorithm is representations of two documents, the output is a binary value indicating the preferred document. as opposed to pointwise this yields a ranking of two documents. On the other hand, all documents have to be compared to each other to get to an ordered list, which can be computationally expensive. Such a system is proposed in [30].

Lotfain and Busso [31] also investigated the possibility of generating a pairwise data set for preference learning based on categorical labels. They developed a framework in which inter annotator agreement is used to assign a preference over a specific axis(angry, happy or sad) in a pairwise ranked data set. They applied Gaussian Process (GP) which slightly outperformed alternative systems.

2.5.3 List-wise

The final ranking of all the documents is used to determine loss of the model. This is state of the art and is competitive with pairwise approaches. Fang et al. [32] propose a recommendation system that is validated on two real world data sets. The first data set consists of product reviews, the second data set consists of movie reviews. The system makes use of list-wise learning-to-rank (LTR) instead of the widely used pointwise ranking. To do so they take the cross entropy between the predicted ranking and the full ground truth ranked list as loss function. The proposed system outperforms pointwise as well as pairwise based systems. Furthermore they use similarity measures to validate individual rankings in order to filter out malicious input.

2.6 Loss Functions

In the different Machine Learning Applications a variety of loss functions are used to evaluate the quality of a regression or ranking output. Furthermore the gradient of this loss function is used to update the model during training. The loss functions used in classification are omitted in this paper, since those loss functions are not applicable for the algorithms developed in this project.

2.6.1 Loss Functions for Regression

Mean Squared Error Loss (MSE)

The Mean Squared Error Loss is a widely used error function. It is closely related to the Sum of Square Error function. In both cases the difference between predicted (p) and target (t) is squared. The squares of every input is summed together. When using Mean Squared Error Loss, this sum is divided by n . This error is sometimes referred to as L2 loss.

$$SE(p, t) = \frac{1}{2} \sum (p - t)^2$$

The factor of $\frac{1}{2}$ is convenient when determining the gradient [33].

Mean Squared Logarithmic Error Loss (MSLE)

$$MSLE(p, t) = \frac{1}{n} \sum (\log(t + 1) - \log(p + 1))^2$$

Mean Squared Logarithmic Error is sometimes preferred over MSE when the target values are widely spread. When using MSLE the error does not increase enormously when a high value target is miss-classified. For example if x_1 and x_2 are predicted as 0.3 and 6 while there respective targets are 0.1 and 5:

$$SE(x_1, t_1) = \frac{1}{2} (0.1 - 0.3)^2 = 0.02$$

$$SE(x_2, t_2) = \frac{1}{2}(6 - 5)^2 = 0.5$$

$$SLE(x_1, t_1) = (\log(0.1 + 1) - \log(0.3 + 1))^2 = 0.02791$$

$$SLE(x_2, t_2) = (\log(6 + 1) - \log(5 + 1))^2 = 0.02376$$

When using SE the error increases enormously between x_1 and x_2 , even though the relative error is smaller for x_2 than for x_1 . Jackner et al. [34] state that logarithmic errors are better suited for relative regression targets.

Mean Absolute Error Loss (MAE)

Mean Absolute Error Loss is the average of the absolute difference between the target and predicted value. This error is sometimes referred to as L1-loss.

$$MAE(p, t) = \frac{\sum |p - t|}{n}$$

A problem with this Loss Function lies with the gradient, which is linear for every value. This means that the gradient is especially large when the error is small. To fix this problem some alterations of MAE are made. Next to that, this loss function is not differentiable when $p-t$ is 0.

Huber Error Loss

Huber Loss [35] is a combination of MSE and MAE. In fact Huber loss is MAE until the error comes below a certain threshold δ . This loss function is less sensitive for outliers but does not have the gradient problem that MAE encounters.

$$H_\delta(p, t) = \begin{cases} \frac{1}{2}(t - p)^2 & \text{for } |t - p| \leq \delta \\ \delta|t - p| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

$$HE = \sum_{n=1}^n H_\delta(p_n, t_n)$$

A problem with this Error function is that it introduces a new parameter δ that has to be tuned.

2.6.2 Loss Functions for list-wise Learning-to-rank

When applying list-wise learning-to-rank a loss functions is needed that determines the loss based on the position a sample takes in the ranking. Xia et al. [36] describe different list-wise loss functions and conclude that the likelihood loss (LL) has the best properties.

$$LL(g(x), y) = -\log P(y|x; g)$$

$$\text{where } P(y|x; g) = \prod_{i=1}^n \frac{\exp g(x_{y(i)})}{\sum_{k=i}^n \exp g(x_{y(k)})}$$

2.7 Hyper Parameter Optimization

When working with machine learning models, many parameters can have an influence on the performance of a model. Parameters such as: number of layers, number of nodes, dropout rate, learning rate and many more. These are all parameters of a neural network model. Next to that the parameters of the audio feature vector can have an influence on the performance of models, such as input format, frame size, frame rate etc. Hyperopt [37] is a python library that can be used to optimize these parameters. As input it needs a search space, in which the different parameters and settings are defined and a loss function which has to be optimized. Hyperopt maps a probability function to the configurations of the search space and the loss function, in order to find the best configuration of hyper parameters for a model given a certain loss function. Bayesian optimization is used to find the best model for a certain loss function.

2.8 Background Ranking Statistics

When comparing machine rankings, there are several standard metrics which are based on precision and recall. Mean Average Precision (MAP) is one of these metrics which is specifically usable to validate the top k ranks of a list. To use this metric, a ground truth is needed to compare the results with. For the machine learning algorithms MAP can be used, but when comparing different annotation techniques another metric is needed.

2.8.1 Similarity Metrics

Fagin et al. [38] describe different ways to compare the top k results of a ranked list without the need of a ground truth. They describe different metrics and measures of similarity. Next to that they describe two metrics of similarity that can be used on the full ranked lists: the permutations.

D = Set of all elements

$$n = |D|$$

S_D = Set of permutations of D

σ = One permutation out of S_D

$\sigma(i)$ = position of i in permutation σ

ρ = set of pairs

$$\rho = \rho_D = (i, j) | i \neq j \text{ and } i, j \in D$$

Kendall' Tau This metric checks the order of pairs of elements i, j . If i and j are in similar order in both permutations: $\overline{K}_{i,j}(\sigma_1, \sigma_2) = 0$. Else, if i, j are in reverse order, $\overline{K}_{i,j}(\sigma_1, \sigma_2) = 1$.

$$K(\sigma_1, \sigma_2) = \sum_{i,j \in \rho} \bar{K}_{i,j}(\sigma_1, \sigma_2)$$

Kendalls tau has the highest value if the two permutations are in reverse order, the maximum Kendalls tau is $\text{Max } K = n(n - 1)$. To normalize Kendall's tau, $K(\sigma_1, \sigma_2)/\text{Max } K$. Kendalls tau is equal to the amount of exchanges needed in bubble sort to convert one permutation in the other.

Spearman's Footrule The Spearmans footrule metric calculates the L1 distance between two permutations.

$$F(\sigma_1, \sigma_2) = \sum_{i=1}^n |\sigma_1(i) - \sigma_2(i)|$$

The maximum value(Max F) of Spearmans footrule metric is $(n^2)/2$ if n is even and $(n + 1)(n - 1)/2$ if n odd, to normalize Spearman's Footrule: $F(\sigma_1, \sigma_2)/\text{Max } F$. A variation of Spearmans footnote, Spearmans rho, is developed to compare the top k results of two permutations.

Where Kendall's Tau looks at the order within a permutation, Spearman's footrule compares the ranks of elements between permutations. This means that if only one sample is ranked differently and the order of the other samples is the same, this could generate a high penalty although one might argue that the two permutations are very similar. The next example explains this problem:

$$\sigma_1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$$

$$\sigma_2 = [1, 10, 2, 3, 4, 5, 6, 7, 8, 9]$$

$$F(\sigma_1, \sigma_2) = 16$$

$$K(\sigma_1, \sigma_2) = 8$$

When using Kendall's Tau, only the element 10 is penalized, when using Spearman's footrule also the following elements give a small penalty.

The above mentioned measures are discussed in different papers. [39] and [40] discuss the transformation from these measures into metrics of correlation coefficients. These metrics calculate a value between -1 and +1, where -1 suggests a negative correlation(rankings are reverse) and +1 suggests a positive correlation(rankings are identical).

The transformation of Kendalls tau into correlation metric Kendall's Rank Correlation Coefficient($KRCC$) is discussed in [39]. $d_{\Delta}(\rho_1, \rho_2)$ denotes the count in the set of pairs which are only in one of the ordered pairs.

$$KRCC = \frac{\frac{1}{2}n(n-1) - d_{\Delta}(\rho_1, \rho_2)}{\frac{1}{2}n(n-1)} = 1 - \frac{2d_{\Delta}(\rho_1, \rho_2)}{n(n-1)}$$

Mukaka et al. [40] discuss Spearman's Rank Correlation Coefficient($SRCC$).

$$SRCC = 1 - \frac{6 \sum (\sigma_1 - \sigma_2)^2}{n(n^2 - 1)}$$

The interpretation of $KRCC$ and $SRCC$ is discussed in [41]. In this paper both coefficients are referred to as r values. The interpretation of this r value is different depending on the research field. For example, an r value of 0.7 or -0.7 (positive or negative correlation) is defined as *strong*, *very strong* and moderate for the respective fields of Psychology, Politics and Medical. Since this project is focused on aggression, the ranges defined for Psychology are used in this project, see table2.1.

Table 2.1: r value interpretation

r value	Relation
1.0 -1.0	Perfect
0.9 -0.9	Strong
0.8 -0.8	Strong
0.7 -0.7	Strong
0.6 -0.6	Moderate
0.5 -0.5	Moderate
0.4 -0.4	Moderate
0.3 -0.3	Weak
0.2 -0.2	Weak
0.1 -0.1	Weak
0.0	Zero

Transitivity

Clinit Davis-Stober describes the transitivity of preference as follows: "To have transitive preferences, a person, group, or society that prefers choice option x to y and y to z must prefer x to z [42]. Furthermore a binary preference is defined as a preference choice between two alternatives, where one is chosen over the other. This can be transformed in the following statement:

$$\forall x, y, z \text{ if } x \succ y \text{ and } y \succ z \text{ then } x \succ z$$

If this proposition holds, this is perfect transitivity. Let matrix A be a preference matrix of elements a, b, c, d, e , where a is the most preferred option and e the least preferred option. If matrix A is perfectly transitive, it would look like this:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Where $a_{ij} = 1$ means that $i \succ j$. When working with real world data, more often than not, perfect transitivity cannot be assumed. To measure the level of in-transitivity, the preference $n * n$ matrix (P) can be compared to a reference $n * n$ matrix (R) using the following formula:

$$\text{Transitivity Error} = \frac{\sum |P - R|}{n^2 - n}$$

In this formula the difference between the preference matrix P and the reference matrix R is divided by the amount of comparisons, excluding the diagonal because no element is compared with itself.

To calculate the weight associated with element x in matrix P , the sum of row i is divided by the sum of the full matrix P .

$$W_x = \frac{\sum_j P_{x,j}}{\sum_i \sum_j P_{i,j}}$$

The sample with the highest weight, will be the highest rank. If the situation of matrix A would be a real world situation with a probability of 0.1 of an in-transitive preference, matrix B could be the preference matrix of this situation.³

$$B = \begin{bmatrix} 0 & 0.9 & 0.96 & 0.91 & 0.93 \\ 0.1 & 0 & 0.91 & 0.84 & 0.87 \\ 0.04 & 0.09 & 0 & 0.89 & 0.88 \\ 0.09 & 0.16 & 0.11 & 0 & 0.9 \\ 0.07 & 0.13 & 0.12 & 0.1 & 0 \end{bmatrix}$$

$$\text{Transitivity Error} = \frac{\sum |B - A|}{5^2 - 5} = 0.101$$

The Transitivity Error is $0.101 \approx 0.1$.

$$W_{x=1} = \frac{\sum_j P_{1,j}}{\sum_i \sum_j P_{i,j}} = \frac{0.9 + 0.96 + 0.91 + 0.93}{10} = 0.37$$

The weight associated with element $x = 1$ is 0.37.

Multiple publications suggest that there can be in-transitivity when working with real world data [43]–[45], but no clear acceptable transitivity error is defined.

2.8.2 Rank Evaluation

(Mean) Average Precision In information retrieval Mean Average Precision is widely used as an evaluation measure. Average Precision (AP) is "the precision at each relevant [sample], averaged over all relevant [samples]" [46]. This measure is mostly used in cases where

³Matrix B is obtained by simulation the experiment 100 times, dividing the preference count by 100. Where the probability of preferring a over b wrongly is 0.1.

a binary relevant score is used(a document is either relevant or not). Mean Average Precision(MAP) is AP averaged over multiple search queries. This is standard practice in Information Retrieval to validate a system. Since this project focuses on one query(aggression), MAP is less relevant.

Normalized Discounted Cumulative Gain (NDCG) When working with continuous relevance scores instead of binary relevance scores, NDCG is a widely used measure in Information Retrieval [47]. Cumulative Gain is the summation over these scores up til a certain index. Discounted Cumulative Gain adds a penalty D for every index further up the list. These measures are mostly used up until a certain point, indicated after the '@' sign.

$$D_r = \frac{1}{\log(1 + r)}$$

$$DCG@p = \sum_{r=1}^p I_r D_r = \sum_{r=1}^p \frac{I_r}{\log(1 + r)}$$

Where I_r is the relevance score of the sample at position r in the ranking.

This measure can be normalized by dividing by DCG of the ideal ranking (IDCG), this measure is called Normalized Discounted Cumulative Gain (NDCG).

$$NDCG = \frac{DCG}{IDCG}$$

State of the Art

3.1 Aggression detection in audio

Kooij et al. [48] describe the development of CASSANDRA, a multi model system for surveillance purposes. The system uses 3D cameras in combination with audio classifiers in order to detect aggression. The audio component classifies audio as Speech, screaming, singing and 'kicking objects'. Furthermore, different audio features are extracted. The classification labels, audio features and video features are combined using a Dynamic Bayesian Network (DBN) in order to determine the level of aggression. The Root Mean Squared Error (RMSE) of the DBN while only using audio features is 0.223, where the full system, using all audio and visual features, yields an RMSE of 0.188.

3.2 State of the Art in Audio Annotation

In emotion annotation, including but not limited to the annotation of audio, the arousal-valence scale [6] is widely used [49], [50]. This scale used to be the state of the art annotation method, but a paradigm shift is taking place. Recent studies [51]–[58] show improved transitivity as well as improved accuracy when using ranked data sets, especially in affective computing.

Yang et al. [52], [53] describe two frameworks in which music is annotated in a ranking for both valence and arousal. In this framework annotators evaluate 8 randomly selected pairs of samples. The samples are compared as in a tournament, which leads to one final top sample. This way of annotating lowers the cognitive load, which makes it easier for annotators. Subjective evaluation of this annotation method, compared to a rating based annotation, showed positive results. On the other hand this only leads to *most* sample and not to a *least* sample, since in the tournament set up only the 'winning' samples are compared to each other in the next round. If this set-up would be extended so that the 'losing' samples are compared to each other as well, the evaluation would also yield a *least* sample.

Yannakakis and Martinez [51] evaluate quality of different annotation methods on inter

annotator agreement. They compared rating annotation with ranking annotation(both on arousal-valence scale) and found that ranking annotation improved inter annotator agreement. They extend their work in [57]. They state different disadvantages of rating based evaluation.

- Inter-personal difference; Two subjects may experience something(i.e. likability of a sound) the same, but rate it differently. One subject might rate the sound as very likable where the other subject rates the sound as extremely likable.
- Ratings are transformed to numerical scales and used as interval, even though the numerical scale is unknown. When subjects rate data through actual labels(i.e. Likert scale), the numerical scale of these labels is unknown, but the annotation is transformed assuming this numerical scale is known. Even when rating is done numerically, these values represent labels. For example ranging from very pleasant to not pleasant at all. Furthermore the assumption is made that this scale is linear, assuming that the distance between fairly pleasant (4) and moderately pleasant (3) is the same as the distance between fairly pleasant (4) and extremely pleasant (5).

On the other hand there is rank-based evaluation, which minimizes the assumptions that a subject has to make and is therefore a fairer evaluation. This is supported by Yannakis and Hallam [56], who compare rating-based and ranking-based evaluation. Within-subject analysis yielded higher in-transitivity for rating-based evaluation.

In an other study Martinez and Yannakakis [54] transform a rating based annotations into classes as well as into a ranking. They train a neural network to classify a sample as either more or less pleasant; by doing so, the model builds a ranking. Next to that they train a neural network to learn a ranking. Both systems are evaluated using Kendall's Tau [59]. The ranking-based network outperformed the classification network.

3.2.1 Hybrid Comparison

Rank annotation is also applied in fields other than emotion annotation. Chevret and Parizet [7] did an experiment where they let a group of 36 people evaluate 12 sounds of closing car doors. The participants were asked to answer the following question: What is the quality of the door?. They had to answer this question on a continuous scale, which was supported with 5 labels ranging from very poor to very high. The 12 sounds were available in one screen, giving the users the opportunity to compare the different sounds. Furthermore the ranking program incorporated an ordering button, which could be used to order the different sounds based on the users quality rating. The authors call this a "mixed procedure", because it is a combination of individual annotation(the continuous scale) and comparison(ordering the samples). The scores of the 36 participants were set against the results of a pairwise comparison test. The scores of the mixed method were comparable to the pairwise comparison, while the participants needed less time to rate the sounds. It has to be noted that the mixed procedure leads to higher variances.

3.3 State of the Art in Automatic Audio Ranking

In audio ranking similar methods as for audio detection are applied. The research of Lotfian and Busso [50] focuses on the comparison of ranking to two different types of Support Vector Machines (SVMs) in a speech recognition task. They investigate if Rank-SVMs can outperform ordinary SVMs and SVMs meant for regression (SVR).

To do so, the SEMAINE¹ database is used, which includes audio data that is annotated with absolute values for arousal and valence. A preference for each pair is determined by comparing these absolute values.

Evaluation is done according to median split. In fact it is a binary classification problem for a specific axis (arousal or valence), this makes it possible to use recall and precision as metrics.

All models use 50 high level descriptors as input. The rank-SVM outperforms both other models in this classification task. For arousal up to 20%, for valence with 7% and 4% for SVM and SVR respectively.

Parthasarathy, Lotfian and Busso [60] elaborate on their own work [50] by extending the work with DNNs. Ranknet is used to create a ranking of arousal, valence and dominance for a spoken conversation corpus. Ranknet outperforms RankSVM as well as the DNN regression model.

Yang et al. [61] use CNNs to predict hit rankings for Western and Chinese pop music. The target value of these songs is based on user listening data and used in a regression task. The hit score is calculated by multiplying a songs play count by the number of distinct users. In both of the subsets there are 10 thousand songs, 8 thousand are used for training, 1 thousand as validation and another 1 thousand as test set.

The study compares shallow networks and deep networks as well as two different types of inputs and the combination of these inputs. One of these inputs is the low-level mel-spectrogram. The other input is generated by using an auto-tagger that extracts 50 high level audio features of a sample. These features are used as input for the different neural networks.

Four metrics are used to evaluate the systems: Recall@100²; nDCG@100(which is similar to recall@100 but also takes the order of the recalled samples into account); Kendall's Tau and Spearman's Footnote.

The results show that deep models perform better than shallow models on the first two metrics. However on the two metrics that take the full ranking into account they score worse. The same goes for the auto-tagged input, this does have a positive effect on the metrics based on recall, but not on Kendall's Tau and Spearman's Footrule.

In two other studies [52], [53] applied learning-to-rank on similar data sets, only these data sets were human annotated. In this project the songs were annotated on the valence-

¹Sustained Emotional coloured Machine-human Interaction Using Nonverbal Expression

²Which is recall of the first 100 returned items.

arousal scale, either as an absolute rating or through paired ranking. Two types of machine learning models were tested on this data set: a regression model and a list-wise learning-to-rank model. The paired-wise learning-to-rank model is omitted in [53] because it is too computationally expensive ($\mathcal{O}(n(n-1))$). In [52] pair-wise learning-to-rank is evaluated but turns out to be highly time consuming.

In both experiments learning-to-rank outperforms the regression model, the best models achieving an accuracy of 73.6% for valence and 87.6% for arousal.³

Li [62] describes the use of point-wise, pair-wise and list-wise learning-to-rank for different applications. He states that there is no model that always outperforms the other two, but list-wise and pairwise usually outperform point-wise models. Furthermore the list-wise model approaches the learning problem more naturally since it takes the ordered ground truth itself as a target variable. The results of Li are supported by Ghanbari [63], who investigates the effect of different loss functions.

3.4 Conclusion of State of the Art

Although rating-based annotation methods are still widely used, different studies show the improvements that can be made using ranking-based annotation methods. This same structure is used in automatic ranking, including but not limited to audio. Learning-to-rank is widely used in information retrieval but has also gained attention in affective computing.

In this research the following ranking-based annotation methods will be investigated:

- Paired comparison. Two types of paired comparison will be investigated to find the most efficient method, while still maintaining transitivity in the rankings. In the first method all samples will be pairwise compared. In the second method the binary search algorithm will be used to build a ranking, so that not all samples will be compared with each other explicitly.
- list-wise comparison. In this method the participant will evaluate more than 2 samples which will be ordered in a ranking. The combination of different lists will yield a final ranking. The results of the list-wise approach will be compared to the two paired comparison methods on time efficiency and transitivity.

The results will yield a balance between time efficiency and transitivity, based on which the best method is chosen and applied to a larger data set.

The larger data set will be used to evaluate a regression based machine learning model as well as a list-wise machine learning model, to investigate if audio samples can be ranked automatically based on aggression.

³To calculate the accuracy of rankings Gamma is used [53]. In this research transitivity, *KRCC* and *SRCC* will be used to evaluate the quality of rankings.

Research Questions

4.1 Research Goals

Through this research we investigate a way to automatically rank audio files based on aggression. To do so a target ranking is needed, which can be used to train a machine learning model that can automatically rank audio files.

In this research two problems are discussed. The first problem is an annotation problem, where audio files have to be ordered based on aggression. In this project a system is developed that can be used to efficiently rank 3000 audio files based on aggression.

RQ1: How can a data set of 3000 audio files be annotated in a ranking in an efficient manner?

To answer this question, some sub questions must be answered.

SQ1.1: What is the level of transitivity in aggression annotation in audio samples? This question must be answered in order to determine if there is a logical order in audio files based on aggression. If humans cannot find an order, then it will not be possible to rank audio files on human annotations.

SQ1.2: How can audio samples be annotated into a ranking based on aggression?

SQ1.3: How efficient are the different types of audio annotation when ranking based on aggression?

By answering these questions on a small subset of the data, an efficient way of annotating the entire data set can be found. When an efficient method is determined the data set will be annotated and can be used for automatic processing.

Another problem in this project is the automatic ranking of audio files. After the data is annotated, different machine learning techniques are evaluated for automatically rank audio files.

RQ2: How can audio samples automatically be ranked based on aggression? To answer this question, different sub questions are answered:

SQ2.1: How can regression be applied to rank audio files based on aggression?

SQ2.2: How can list-wise learning-to-rank be applied to rank audio files based on aggression?

The results of both methods are compared in order to find a suitable application for Sound Intelligence.

Research Methodology

5.1 How can 3000 audio files be annotated in a ranking in an efficient manner?

The data set supplied by Sound Intelligence consists of roughly 3000 audio samples. To pairwise compare all files is not feasible, therefore two frameworks for building the ranked data set are proposed and tested.

5.1.1 Baseline

The baseline ranking is build using pairwise comparison, which is the most discriminating way of comparing audio samples into a ranking. For more information see chapter 2.1.2. This ranking is build using a pairwise comparison, in which the subjects have to compare the full matrix. In the first experiment 12 samples are evaluated, which lead to 66 comparisons($N(N - 1)/2$) for the baseline.

5.1.2 Binary Search

The first tested method is based on paired comparison and binary search explained in 2.1.3. The comparison is based on aggression. if the subject evaluates sample x more aggressive than sample y , the algorithm will move up in the list. If the comparison is evaluated the other way around, the algorithm will move down in the list.

5.1.3 Hybrid Comparison Insertion

This method is based on the hybrid comparison method explained in 3.2.1. It is not possible to let subjects make an ordered list of all the audio samples. In previous research participants had to compare at most 12 samples at a time. To use this technique on a bigger data set, the existing ordered list is split into 12 parts. Of each part one sample will be used in a selection that has to be ranked by the participants. To this selection an unranked sample x

is added. The samples in between which x is ranked determine the new ordered list which will again be split into 12 parts. This process will be repeated until x has a final position.

In order to simulate this ranking method while answering RQ1, the participant has to rank two lists of 7 samples in the first experiment. In these samples there will be two anchor samples, which occur in both lists. The anchors are obtained from the baseline. The lowest ranked sample and the highest ranked sample from the baseline will be used as anchors for the multi-list method.

Audio Samples 12 audio samples are selected randomly from the labels human speech, raised voice and aggression. Of all three labels, four samples will be selected.

Participants The participants are naive subjects, in other words: not trained in audio annotation. The participants are students present at the University of Twente at the moment of the experiment. For each type of comparison, 25 subjects will take part in the experiment.

Analysis The results of the first experiment are validated based on transitivity and time efficiency.

Transitivity

Within every method, the following measures are compared. These measures should indicate if different participants agree with each other when using the same annotation technique.

- transitivity(as explained in 2.8.1) within group
- Mean KRCC/SRCC within group (one vs rest)

The above two measures are useful to see the level of agreement within one method. If this differs over the ranking methods, this can be an argument to choose one over the others. Furthermore the methods are compared to the baseline results, which counts as a groundtruth for this experiment.

- transitivity compared to baseline.
- *KRCC/SRCC* compared to baseline

Literature suggests that a full paired comparison yields the most transitive and discriminating results. This analysis should find a more efficient method that achieves similar results as the baseline.

Time spend

time cost is rated using the following measures:

- Mean time spend(for 12 samples)
- Mean time spend per comparison
- Amount of comparisons needed to rank full and partial data set

- Time needed to annotate full and partial data set.

Based on the above measurements a simulation of the annotation process is done. The most important measurement is based on the quality of the ranking. To evaluate the quality, transitivity, $KRCC$ and $SRCC$ is used. Based on the values of these measures, the most discriminating method is selected.

If these values are similar for multiple methods, time efficiency is used to select the best method. First, the amount of comparisons needed to annotate 3000 samples is calculated for each individual method. The amount of comparisons multiplied by the time spend per comparison is used to give an indication of the time needed to annotate the full data set. Based on the combination of the quality of the ranking and the time needed to annotate a full ranking the most appropriate method is selected.

Ranking Simulation

When the most efficient ranking method is found, a simulation of the ranking process is done. This is used to find an indication of the expected in-transitivity for specific batch sizes, count of annotators and rank aggregation methods.

The simulation is done using a range of integers according to the different batch sizes. This range of integers is ranked using the most efficient ranking method. The simulation is done for different error rates, which defines the probability of a mistake: 0.05, 0.1, 0.15. A mistake is defined as two samples that are not ordered in descending, but in ascending order.

The simulation is done using the following parameters:

- Batch size: 10, 50, 100, 200, 300
- # annotators: 3, 5, 10
- Rank aggregation methods: Mean, Mode, Median

Furthermore, a simulation of the extension of the first batch ranking is done. In this simulation a start batch of size 10, 50 or 100 is generated using the best rank aggregation method and optimal annotator count. This ranking is used to add new files 1 by 1, based on the agreement of multiple annotators.

5.2 Automatic Ranking of Aggression

5.2.1 Data

Sound Intelligence supplied a data set of recordings that they use to train their current aggression detector. These recordings are from a variety of sources. The bulk of the recordings are from a database called A. Next to database A there are two other sources that supply a fair amount of recordings, called database B and database C. Next to these 3 main sources there are recordings from other small databases.

The recordings in database A are recorded in healthcare institutions, the recordings in database B are recorded in prisons, the recordings in database C are recorded in an expertise centre for epilepsy. The recordings within a database can be very similar. Furthermore it is difficult to compare recordings from outside of a specific database, if most of the other samples are from the same database. Therefore a maximum number of recordings from these three main databases is set. The diversity in sources yields a diversity in audio fragments, which improves the generalizability of machine learning models.

The full data set consist of 2427 audio samples drawn from the Sound Intelligence databases. The data is labelled by a group of trained annotators. The annotators select the part of an audio sample in which an event occurs and giving this part the appropriate label. The selected part of the audio is saved with the label. The labels used in this research are *Voice*, *Raised Voice* and *Screaming*. The samples are drawn using the following conditions:

- A maximum of 1000 files is randomly drawn from each of the following categories: *Voice*, *Raised Voice* and *Screaming*, since these categories are relevant for aggression detection. Furthermore, these categories are used in the current aggression detection model.
- A maximum of $\frac{1}{3}$ samples in each category may come from database A.
- A maximum of $\frac{1}{3}$ samples in each category may come from database B.
- A maximum of $\frac{1}{3}$ samples in each category may come from database C.
- The other samples come from diverse sources.

A final ranking is obtained by applying the annotation method developed answering Research Question 1, 5.1.

5.2.2 Data Analysis

The rankings of different annotators are compared to determine if a final ranking can be obtained that can be used to train and validate machine learning models. This final ranking should be acceptably transitive. It can be assumed that there is some degree of in-transitivity when working with real world data. Furthermore, the individual rankings should be positively correlated with each other. This indicates that the different annotators agree on the order of the recordings based on aggression. Therefore the rankings are individually compared with each other on KRCC, SRCC and in-transitivity. These measures are explained in 2.8.1.

Furthermore, the annotations are used to build four matrices and used to compare the individual annotations.

- Weighted Matrix, this is the matrix obtained by transforming every individual ranking into one paired matrix. In this matrix every position M_{ij} is the probability that $i \succ j$.

- Linear Matrix, this is the reference matrix obtained by the ranking of the Weighted matrix. Note that this matrix is perfectly transitive.
- Median Matrix, this is the paired matrix that follows from the final ranking, obtained by aggregating the individual rankings using the median position of every sample.
- Median Matrix(equal allowed), this is the paired matrix similar to the median matrix only now shared positions are allowed. If two samples share a position, the position in the matrix takes a value of 0.5.

More information on these matrices can be found in 2.8.1.

Next to the individual rankings, the individual sample will be analyzed. It is assumed that the samples that are difficult to rank, will be ranked far apart over the different rankings. Based on the standard deviation of individual samples in the different rankings, a list of difficult to rank samples is specified.

The median ranking are used as a target for the machine learning models. For the regression model, the position of a sample is divided by the length of the ranking. This yields a value between 0 and 1 which can be used as a target value. For the list-wise learning-to-rank approach, a scoring function per sample is optimized, which gives a high value to relevant samples and a low value to irrelevant samples, so that the samples are ordered from most aggressive to least aggressive.

5.2.3 Audio Ranking using Machine Learning

When applying fully connected Neural Network architectures it is important to supply the architecture with an input of the same size for every data point. Figure 5.1 shows the architecture of a fully connected neural network with an input layer, 1 hidden layer and an output layer. The size of the input in this example is 3 nodes. This means that every data point that goes through this network, should be represented as a vector of length 3.

The data set used in this project consists of audio recordings of different lengths. All recordings are sampled at the same frame rate. Therefore the data points of this data set all have different lengths, and should be processed before they can be supplied to an NN architecture. The problems that occur in the processing of unequal length audio samples are explained in 2.4.4.

Furthermore, the recordings will be annotated at clip level, which means it is unclear in which part of the recording the actual aggression occurs. This might not be a problem because the recordings are already trimmed to the event of the original label (*Voice*, *Raised Voice* or *Screaming*), but the new annotation would still be classified as clip level annotation. More information on weakly labeled data can be found in 2.4.3.

To deal with varying length data input, all samples are split into data frames. Every frame is supplied to the network individually or in sets, with the clip level annotation as the target

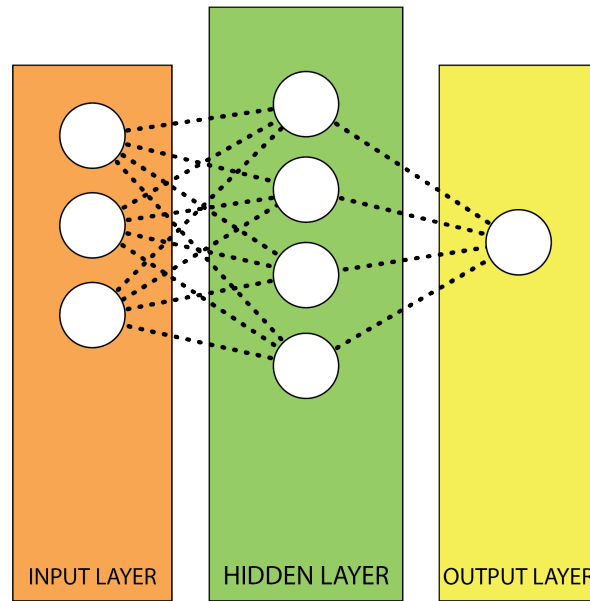


Figure 5.1: Fully Connected NN with an input layer of length 3.

value. The predicted value of the different frames of a specific recording are averaged to predict the value of a full recording. It is assumed that most information about aggression will be in the frames with the most energy. Therefore the predicted value is weighted by the energy of that sample to determine the average predicted value for each recording. This is done to deal with the weak labels. These values are compared to the unweighted predicted values for each recording to see if this leads to improvements in performance.

The wave recordings are sampled at a frame rate of 8000Hz and transformed into Log-Mel-Spectrums(LMS) of different frame sizes. These frame sizes are optimized during hyper parameter optimization. Next to the LMS of the current frame, the feature vector includes the LMS of previous frames, this is added to the current frame. The amount of frames in one feature vector is optimized during hyper parameter optimization.

The error function of the regression model is based on the distribution of the target values. As explained in 2.6.1, Mean Squared Loss is useful if the distribution of target values approaches the normal distribution, while Mean Squared Logarithmic Loss is more resistant to outliers.

The error function of the list-wise model is the Likelihood Loss explained in 2.6.2.

5.2.4 Performance

To determine if it is possible to automatically rank audio samples based on aggression, a set of 300 samples is manually annotated using the most efficient method(selected through the first experiment) and used as input for different machine learning architectures. This annotated set is split into two sets, a training set of 250 samples and a test set of 50 samples. The training set is used to configure different architectures. This is done by applying 5-fold

cross validation: The set is split in five parts of 50 samples, a model is trained on four of these parts and tested on the remaining 50 samples. This is done five times, so that each part is used for validation. The validation scores are averaged to give a more generalized score for each model. Based on the validation scores the best models are selected and trained on the full training set. These models are then tested on the test set to determine the best model. The results will be analyzed on different aspects:

- Loss results. The results of the loss functions will be used to tune the parameters of the different models. Although MSE is not used as a loss function in the list-wise systems, it is calculated as a metric.
- KRCC/SRCC, these values can be compared to the results of the annotated rankings.
- Difficult to rank samples. This set will be used to see if the system improves if those samples are removed.
- Sensitivity test. First the models will be trained on all available training data. When the best models are selected, they will also be trained on less data to see what effect this has on the performance.

5.2.5 Hyper Parameter Optimization

To find the best model; hyper parameter optimization is done using the python library Hyperopt(more information in 2.7). The search space can be divided into two parts, the feature search space and the model search space. Furthermore, two types of models are optimized, a Fully connected neural network(Dense) and a Convolutional Neural Network(CNN). As a loss function, $1 - \text{SRCC}$ is used since this would optimize a model that returns the ranking closest to the ground truth ranking. This measure only takes the order of samples into account and ignores the target/predicted value. Furthermore, it is a measure calculated over the final clip level ranking instead of the value per frame or frame set.

A fully connect neural network is included because it is the most basic structure of a neural network. Furthermore, other networks are based on or include parts of this type of structure. Figure 5.2 shows the structure of the Dense network. The length of the input depends on two variables that are optimized through hyper parameter optimization: The frame size and frame count. The input consist of the LMS of the current frame, which can be 256, 512, 1024 or 2048 long. This frame is extended by adding the LMS's of previous frames, this can be done for 1 up to 32 frames(the maximum frames is lower when the frame size increases). The output layer always consists of one node with the sigmoid activation function. The hidden layer consists of x layers with y nodes. x and y are selected using hyper parameter optimization, and are chosen from: x layers from an integer value between 1 and 10; and y nodes per layer chosen from: 10, 50, 100, 200, 350, 500. All layers include a dropout value uniformly chosen between 0 and 1. The activation function for all layers except the output layer are chosen from the following set: Relu, Tanh, Sigmoid or Selu. Next

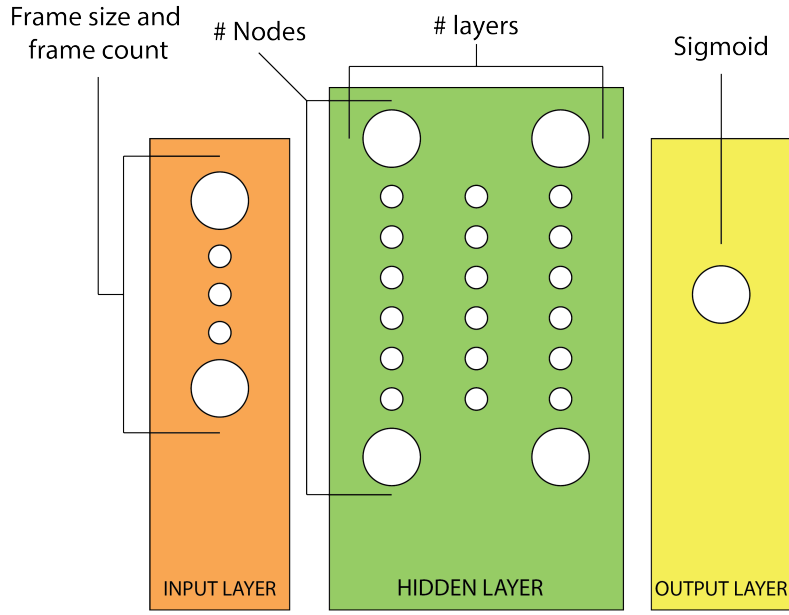


Figure 5.2: Dense Network structure

to that a regularization is applied to these layers, either L1 or L2 regularization is applied with a factor log-uniformly chosen from the range of 10^{-5} up to 10^{-1} . The learning rate is optimized as well, log-uniformly chosen from the range of 10^{-5} up to 10^{-1} .

The convolutional neural network is included because the data used in this project is annotated at clip level. Because this is the case, it is unclear in which part of the audio sample the annotated level of aggression occurs. The convolutional structure can learn local features of the input, and ignore the noise in the rest of the data. Figure 5.3 shows the structure of the CNN. Again the frame size and frame count are optimized similar to the dense network. Only in the CNN the frames are not extended in a one-dimensional array. The LMS's of the different frames are stacked in a second dimension, making it possible to apply 2D convolution. After every convolutional layer a max-pooling function is applied. The window size of this max pooling layer is optimized in the following range: 1-4. The filter size for the convolutional layers are optimized out of the following range: 2-6. The number of filters is optimized using 32, 64, 128, 256 and 512 as options. After the convolutional layers a flatten layer is added. This layer is followed by a number of fully connected layers optimized similar to the layers in the dense network. Learning rate, dropout and regularization are optimized in the same manner as for the dense network.

Of both architectures a regression and a list-wise model is optimized, yielding 4 different models: Dense regression, Dense list-wise, CNN regression and CNN list-wise.

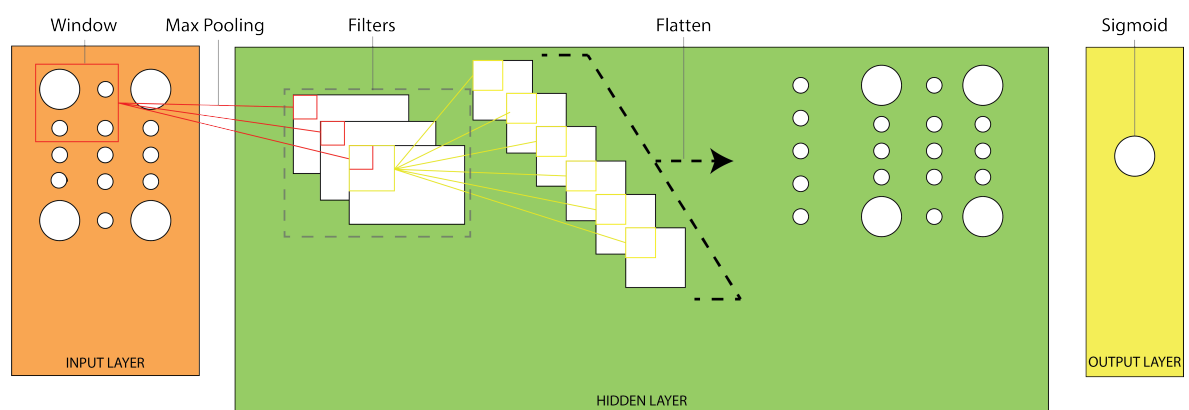


Figure 5.3: CNN structure

Results: Efficient Rank Annotation

6.1 RQ1: How can 3000 files be efficiently annotated?

As explained in the methodology 5.1 different annotation methods are compared in order to find the most efficient and accurate method for the ranking of audio samples. The following systems are tested:

- Baseline system; 25 participants carried out a pairwise comparison of 12 audio samples. This led to 66 comparisons.
- Binary system; 50 participants did a pairwise comparison of 12 files, only now the ranking is build using insertion sort, reducing the amount of comparisons. The system assumes transitivity, this can be applied.
- list-wise system; based on the system of Chevret and Parizet [7]. 25 participants evaluated 12 sounds in a list. For every sound they could set a slider value of aggression. The system would sort and they could check the ranking
- Multi-list system; Similar to list-wise, only now two lists of 7 files(including 2 anchors) had to be made.

Table 6.1 shows the in-transitivity as well as Mean *KRCC* and mean *SRCC* for the four different methods. The in-transitivity score is calculated by comparing the normalized pair matrix to the reference matrix of the ground truth as explained in 2.8.1. This reference matrix

Table 6.1: Transitivity results RQ1

	In-Transitivity	Mean <i>KRCC</i>	Mean <i>SRCC</i>	Mean Time Spend
Baseline	0.150	0.715	0.845	793.25
Binary	0.146	0.707	0.831	360.53
list-wise	0.157	0.682	0.809	209.29
multi-list	0.169	0.631	0.770	227.82

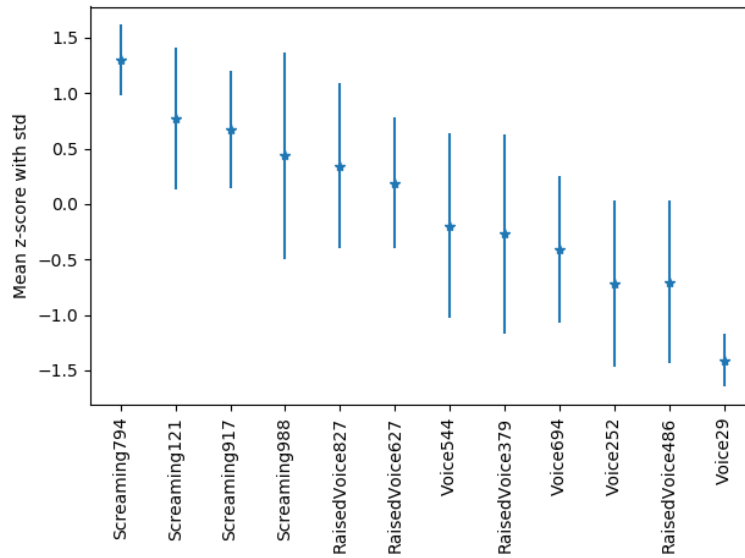


Figure 6.1: Z-scores per sample(x-axis) for the list-wise system

is a perfectly transitive matrix related to the ranking that was extracted from the ground truth experiment(baseline).¹

The mean *KRCC* and mean *SRCC* are calculated as the mean value over all individual rankings compared to a ranking of the combination of all other rankings(one vs rest). The mean time spend is the average time in seconds, that was needed to rank 12 samples.

Figure 6.1 shows the z-scores² and error bars per sample for the list-wise system. Figure 6.2 shows these scores for the multi-list system. The z-scores are the normalized values obtained from the sliders in respectively the list-wise and the multi-list system. The values are normalized by subtracting the mean and dividing by the standard deviation.

The z-scores extracted from the results of the list-wise system seem to be uniformly distributed. The multi-list system does not show this result as clearly, but the samples on the x-axis were not presented at the same time. Figure 6.3 shows the z-scores for each position in the multi-list system. These values seem to be linearly distributed.

6.1.1 Transitivity

The *KRCC* and *SRCC* of the baseline(Table 6.1) as well as the binary system show a strong positive correlation over the different annotators, indicating that they agree on the ranking of the 12 samples. The *KRCC* of the list-wise and multi-list system only indicates a moderate correlation. The *SRCC* of these systems indicates a strong correlation.

¹The value of the baseline in-transitivity is not 0, because the normalized matrix of the baseline is compared to a perfect transitive pair matrix(only including zeros and ones).

²z-scores are used to normalize the samples around the sample mean in terms of STD, as explained in A.1

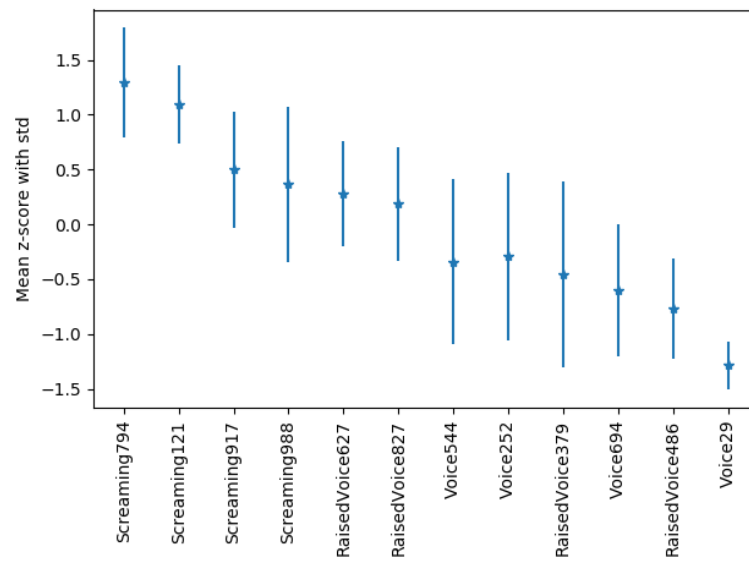


Figure 6.2: Z-scores per sample(x-axis) for the multi-list system

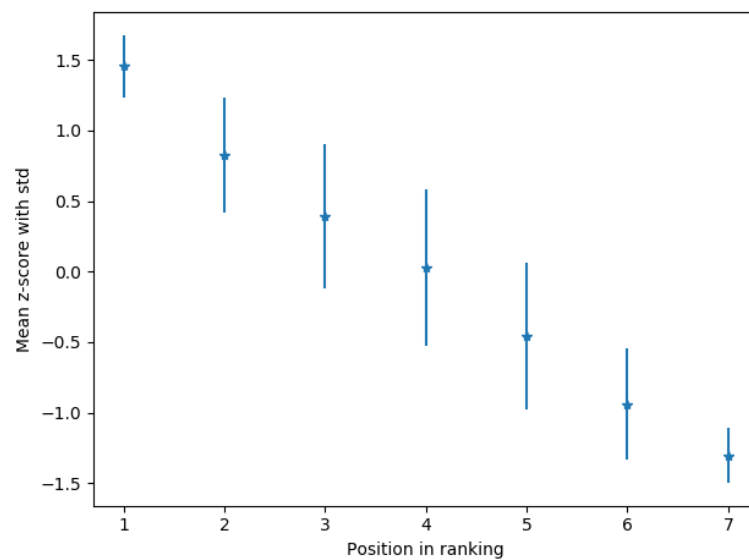


Figure 6.3: Z-scores per position(x-axis) per ranking window in the multi-list system

6.1.2 Time Efficiency

The mean time spend to annotate 12 samples can be found in table 6.1. The baseline comparison took the most time to complete, with an average time of approximately 13 minutes to arrange 12 samples into a ranking. The binary method took approximately 6 minutes, the multi-list system took approximately 4 minutes and the list wise approach is the most time efficient, with approximately $3\frac{1}{2}$ min.

6.1.3 Conclusion Exploration of Annotation Methods

The paired comparison shows a strong correlation (*KRCC* and *SRCC*) over the different annotations. Furthermore it shows a value of 0.15 in-transitivity rate, which is acceptable since real world data is used. When judging emotions, disagreement over different annotators can be expected. Based on these results it can be concluded that there is an order in the audio samples based on aggression.

The binary comparison approach shows the lowest in-transitivity value, as well as the strongest correlation on both *KRCC* and *SRCC*. Therefore this is the most discriminating method of the methods feasible to use for a bigger data set.

The z-score values (figure 6.1) obtained from the list-wise approach seem to be uniformly distributed. In the multi-list system the shape looks stepwise. It has to be noted that only half of the samples (minus the most and least aggressive samples) were presented in one screen. Figure 6.3 shows the z-scores for each position in the compare frames of the multi-list system. It seems that the annotators distributed the values on the scales uniformly between 0 and 1, which would mean that two files cannot be ranked based on this value if they were not presented in the same list. This is in line with the results of [9].

The list-wise system is the most time efficient manner, although a problem occurs. For 12 samples it is possible to order the files in one list, but this is impossible for 3000 samples. Therefore the multi-list system is needed. The binary system took significantly more time to arrange 12 samples than the list-wise alternatives, approximately 6 minutes against approximately 3 to 4 minutes. On the other hand, when applying the binary system more comparisons had to be made, in the worst case 33 comparisons. When applying a list-wise approach, 12 comparisons can be made at a one time. This leads to an average time per comparison of 12.41 seconds for the binary approach and 17.44 seconds for the list-wise approach. This information is used to simulate the annotation of 3000 files, as explained in the next section.

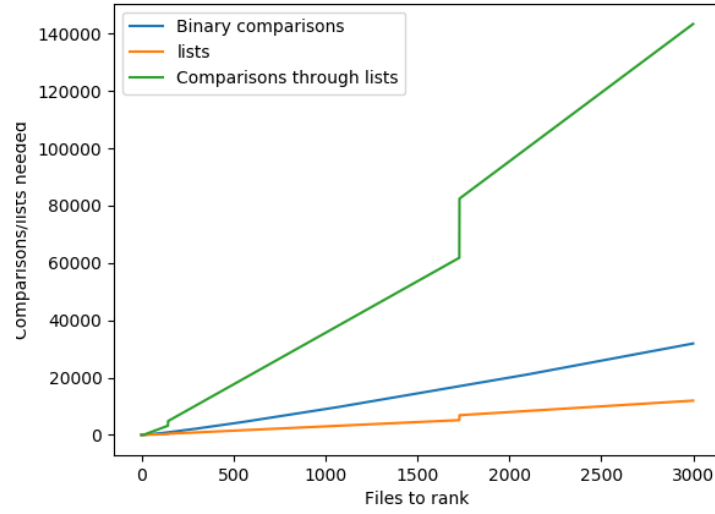


Figure 6.4: Number of comparisons needed per system

6.2 Expected time for 3000 samples

When applying the binary system, $\log n$ comparisons are needed to add one sample to a ranking of length n . To calculate the amount of comparisons needed to build a ranking of length n when there is no initial ranking, the following formula applies:

$$N_{\text{Binary worst case}} = \sum_{i=1}^n \log_2(i)$$

When applying a list-wise system the participant makes a ranking of x files, narrowing the search field to $\frac{1}{x}$ th of the ranking of length n . this leads to $\log_x n$ lists to add one file to the ranking. To calculate the amount of lists needed to build a ranking of length n while applying a list-wise system, the following formula applies:

$$N_{\text{list-wise worst case}} = \sum_{i=1}^n \log_x(i)$$

Furthermore, the amount of lists has to be multiplied by the amount of files in that list to compute the number of comparisons needed. For the simulation a list of 12 items is used.

figure 6.4 shows the amount of comparisons needed to annotate rankings of different batch sizes(x axis). It is clear to see that the amount of comparisons needed when applying the listwise approach is a lot higher compared to the binary approach. Therefore the binary approach is the most time efficient option. When annotating 300 samples this would take 8 hours. When annotating 3000 samples this would take 115 hours.

The most time efficient method is binary comparison. This is also the method in which the participants agreed most with each other. Therefore this method is chosen as most efficient method to annotate audio samples in a ranking based on aggression.

Since it would still take one person 115 hours to annotate the full data set, a subset will be annotated. It seems feasible to annotate 300 samples; this would take 8 hours per person. To decide on the exact annotation method different simulations are done which will be explained below.

6.2.1 Simulate annotation

To determine an indication of the expected in-transitivity for specific batch sizes, number of annotators and rank annotation methods a simulation of the annotation process of 300 samples is done.

The simulation is done using a range of integers according to the different batch sizes. The goal is to build an ordered list by applying the binary comparison algorithm on a list of integers. Instead of aggression, the integers are compared based on their value. The error rate determines if this comparison is answered correctly, or a mistake is made. The simulation is done for different error rates, which define the probability of a mistake: 0.05, 0.1, 0.15.

The simulation is done using the following parameters:

- Batch size: 10, 50, 100, 300
- # annotators: 3, 5, 10
- Rank aggregation methods: Mean, Mode, Median

Figure 6.5 and figure 6.6 summarize the information in Appendix G. These figures show the results of these simulations individually. The plots show the average values of $KRCC$ (Figure 6.5) and $SRCC$ (Figure 6.6). The blue line indicates the average value of the individual simulated annotations, the blue field covers the minimum and maximum values belonging to these annotations over the following error rates: 0.05, 0.1, 0.15.

The other three colors relate to the three different rank annotations methods, yellow: Mean position of the different annotations. Red: Median position of the different annotations. Green: Mode position of the different annotations.

As can be seen the median rank aggregation yields the highest $KRCC$, closely followed by mean rank aggregation. On $SRCC$ mean and median rank aggregation yield similar results. On both measures mode rank aggregation scores lower than the other two.

Next to that the Mean $KRCC$ and $SRCC$ of the final ranking is calculated while using 3 different numbers of annotators. The values are calculated for 3, 5 and 10 annotators. The median ranking of a sample is used to construct the final rankings. The best case is calculated using a 0.05 mistake probability. The worst case is calculated using a 0.15 mistake probability, the results are shown in table 6.2.

Next to the simulation of annotation, in which every annotator first completes a full ranking before the rankings are combined, another system is simulated. In this system the

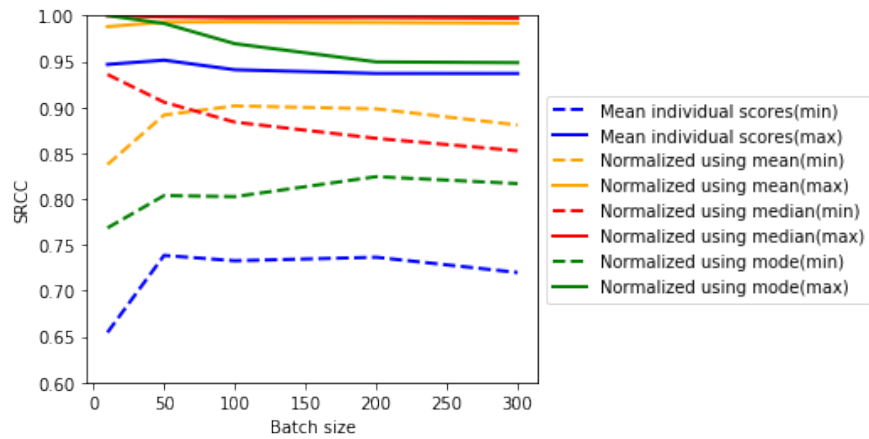


Figure 6.5: SRCC values for the different simulations

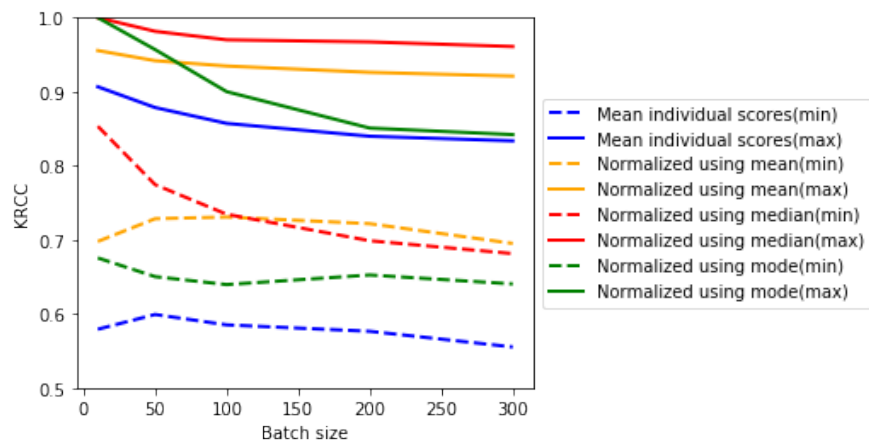
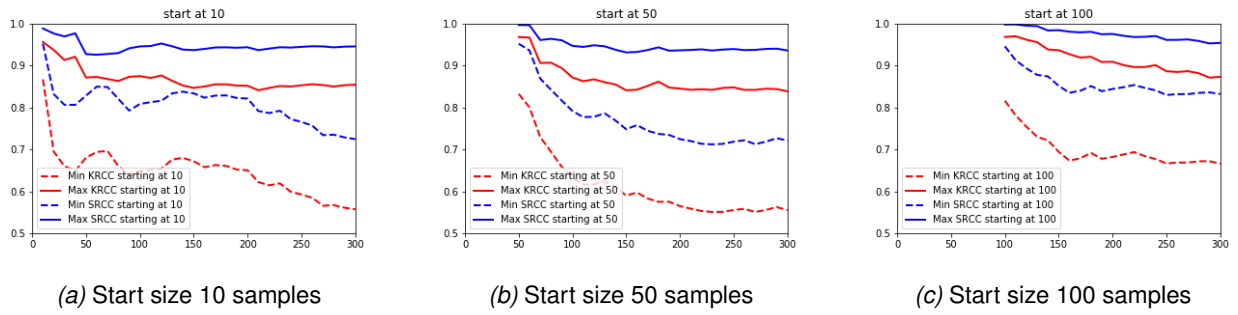


Figure 6.6: KRCC values for the different simulations

Table 6.2: KRCC and SRCC for the different annotator counts.

# annotators	KRCC		SRCC	
	Best	Worst	Best	Worst
3	0.9085	0.6814	0.98	0.8525
5	0.9383	0.7404	0.9915	0.9049
10	0.961	0.8248	0.9969	0.9576

annotators build on a final set together. To do this, a starting set is needed. To build this starting set every annotator builds a ranking of 10, 50 or 100 samples. These rankings are combined into one ranking. After this files are added one by one based on the votes of different annotators. In this case every annotator compares one file to the existing list, if every annotator has compared the sample, it is added to the ranking and a new sample is selected. This method is simulated with starting size 10, 50 and 100 samples, the results are shown in figure 6.7.

**Figure 6.7:** 300 samples

6.2.2 Conclusion

The results of the simulations show that the final ranking will be most similar to the ground truth if every annotator ranks the full subset of 300 samples and rank aggregation is applied using the full set of every annotator. Next to that the median is chosen as optimal aggregation method, since it ignores outliers. If the median of two samples is equal, the mean value is used to determine the position.

Results: Data analysis

The annotations of four different annotators are compared on transitivity, KRCC and SRCC. The different annotations are aggregated into final ranking, which are also compared on transitivity, KRCC and SRCC.

7.1 Individual *KRCC* results

Table 7.1: *KRCC* for different Annotators

	Anno. 1	Anno. 2	anno. 3	anno. 4
Anno. 1	1.0	0.574	0.505	0.509
Anno. 2		1.0	0.510	0.498
Anno. 3			1.0	0.476
Anno. 4				1.0

KRCC is computed for every combination of two annotators, the results are shown in table 7.1. The mean *KRCC* over the different annotators is 0.511, which indicates a moderate positive correlation.

7.2 Individual *SRCC* results

Table 7.2: *SRCC* for different Annotators

	Anno. 1	Anno. 2	anno. 3	anno. 4
Anno. 1	1.0	0.78	0.689	0.695
Anno. 2		1.0	0.703	0.688
Anno. 3			1.0	0.659
Anno. 4				1.0

SRCC is computed for every combination of two annotators, the results are shown in table 7.2. The mean *SRCC* over the different annotators is 0.702, which indicates a strong

positive correlation.

7.3 In-transitivity versus the different normalized rankings

Table 7.3: In-transitivity versus the different normalized rankings

	Anno. 1	Anno. 2	anno. 3	anno. 4
Linear matrix	0.141	0.146	0.158	0.159
Weighted Matrix	0.176	0.177	0.189	0.19
Median Matrix	0.136	0.137	0.160	0.164
Median Matrix (Eq)	0.136	0.138	0.160	0.164

Four different matrices are used to compare the individual results of the annotators: The weighted matrix, the linear matrix, the median matrix and the median matrix with equal positions. These matrices are explained in chapter 5.1.

The mean in-transitivity value of the linear matrix is 0.151, of the weighted matrix 0.183, of the median matrix and the median matrix with shared positions the in-transitivity value are 0.149 and 0.15 respectively.

To further analyze the data, a ground truth ranking is generated. The groundtruth ranking is based on the median position of the sample. The simulation of the experiment suggested that this is the best rank aggregation method. This is supported by the fact that this matrix has the lowest mean in-transitivity value.

7.4 *KRCC* and *SRCC* against ground truth

Table 7.4: *KRCC* and *SRCC* against ground truth

	Anno. 1	Anno. 2	anno. 3	anno. 4
<i>KRCC</i> ranking	0.718	0.708	0.685	0.682
<i>SRCC</i> ranking	0.896	0.897	0.867	0.861

The mean *KRCC* value of the final ranking is 0.698, the mean *SRCC* value of the ranking is 0.88. both indicating moderate to strong correlation.

7.5 Qualitative Feedback From Annotators

The annotators sometimes encountered difficulties when comparing the samples. They indicated that some samples are very similar to each other which can make it difficult to determine which sample is more aggressive. On the other hand, some samples were found to be irrelevant when it comes to aggression in cases where there is undefinable human noise in the audio sample.

7.6 Conclusion

As explained in 7.4 there is a moderate to strong correlation in the rankings. According to the literature a moderate correlation can be assumed if the correlation value is between 0.4 and 0.6 and a strong correlation can be assumed if the correlation value is between 0.7 and 0.9. Based on the mean *KRCC* score(0.698) a moderate to strong correlation exists between the final rankings and the individual annotations.

Based on the mean *SRCC* score(0.88) a strong correlation exist between the final ranking and the individual annotations.

The in-transitivity values explained in 7.3 indicate that there is a level of in-transitivity in the rankings, which can be assumed when working with real world data.

The ranking seem to be usable for machine learning applications because there is a moderate to strong relation between the individual annotations and the final ranking. The simulation of the annotation process showed that the median ranking yields the best groundtruth, therefore the median ranking is used as target value for both the regression and the list-wise machine learning approach.

Results: Machine Learning

This part can be split into two separate methods of ranking, regression and list-wise ranking. In the first method: regression, neural network models are applied to predict a value between 0 and 1 for each input data. These architectures make use of the Mean Squared Error Loss function, explained in 2.6.1. The target is a linear value, based on the position of a sample divided by the length of the data set. In this research the target of x is the position of x in the median ranking divided by 300. This is done because the ranking is build using binary comparison and no information about the target distribution is known, therefore uniform distribution is assumed.

8.1 Hyper parameter optimization

8.1.1 Regression

The best parameter settings given the the search space for the dense model can be found in table 8.1, the best settings for the CNN can be found in table 8.2. In both cases the Adam optimizer is used.

Dense			
input	values	model	values
mels	128	activation	tanh
frame size	512	regularization	L2(0.0000138)
frame count	8	learning rate	0.00374
		dropout rate	0.3613
		number of layers	3
		number of nodes	50

Table 8.1: best parameter settings Dense Network.

CNN			
input	values	model	values
mels	64	activation	relu
frame size	512	regularization	L2(0.00455)
frame count	14	learning rate	0.0000305
		dropout	0.3318
		number of filters layer 1	256
		number of filters layer 2	32
		filter size 1	3
		filter size 2	4
		pool size1	3
		pool size2	3
		nodes	20
		layers	2

Table 8.2: Best parameter settings CNN Network.

8.1.2 List-wise

The best parameter settings given the the search space for the dense model with list-wise loss function can be found in table 8.3, the best settings for the CNN with list-wise loss function can be found in table 8.4. In both cases the Adam optimizer is used.

Dense			
input	values	model	values
mels	32	activation	selu
frame size	1024	regularization	L1(0.0775)
frame count	1	learning rate	0.00306
		dropout rate	0.7057
		number of layers	5
		number of nodes	50

Table 8.3: best parameter settings Dense Network with list-wise loss function.

8.2 Results five fold cross validation

8.2.1 Regression

Table 8.5 show the results of Dense and CNN when trained on exactly the same data set. The data set is loaded, shuffled and split in 5 parts. Each of the 5 parts is used as a validation set, on which $KRCC$ and $SRCC$ is calculated. The other 4 parts are used for training. The same train/validation set is used for the list-wise models, which are explained in 8.2.2. The

CNN			
input	values	model	values
mels	64	activation	sigmoid
frame size	256	regularization	L2(0.0000159)
frame count	17	learning rate	0.000316
		dropout	0.487
		number of filters layer 1	128
		number of filters layer 2	128
		filter size 1	3
		filter size 2	6
		pool size1	1
		pool size2	2
		nodes	50
		layers	3

Table 8.4: Best parameter settings CNN with list-wise loss function.

Table 8.5: *KRCC*, *SRCC* and MSE for Dense and CNN on the same data set using 5-fold cross validation using regression. The best results are made bold.

fold n	Dense Regression			CNN Regression		
	<i>KRCC</i>	<i>SRCC</i>	MSE	<i>KRCC</i>	<i>SRCC</i>	MSE
1	0.5543	0.7613	0.0529	0.5641	0.7608	0.0602
2	0.5200	0.7352	0.0533	0.5265	0.7237	0.0372
3	0.4433	0.6075	0.0789	0.5331	0.7356	0.0349
4	0.5478	0.7603	0.0512	0.5902	0.8064	0.0311
5	0.4890	0.6910	0.0576	0.4661	0.6549	0.0501
mean	0.5109	0.7111	0.0588	0.5360	0.7363	0.0427

last row of the table indicates the mean value per measure. In table 8.5 the highest value for each measure for each fold is in bold. For MSE the lowest value is made bold, since this indicates the best result.

When calculating the significance of both models over the 5-fold cross validation, paired student t-test¹. is used. None of the above mentioned measures indicated significant different means. The p-values for *KRCC* can be found in table 8.8, the p-values for *SRCC* can be found in table 8.9 and the p-values for MSE can be found in table 8.7.

8.2.2 List-wise

Table 8.6 show the results of Dense and CNN(list-wise models) when trained on exactly the same data set, both networks are trained on exactly the same folds as the regression

¹T-test is used to determine significance when comparing two means drawn from a dependent sample group. T-test is explained in A.2

Table 8.6: $KRCC$, $SRCC$ and MSE for Dense and CNN on the same data set using 5-fold cross validation using list-wise. The best results are made bold.

fold n	Dense list-wise			CNN list-wise		
	$KRCC$	$SRCC$	MSE	$KRCC$	$SRCC$	MSE
1	0.5265	0.7149	0.0797	0.5592	0.7713	0.1301
2	0.4008	0.5731	0.0734	0.4253	0.6352	0.1139
3	0.5706	0.7678	0.0780	0.4412	0.6115	0.0885
4	0.3681	0.5215	0.0552	0.5020	0.7026	0.0899
5	0.3420	0.5038	0.0593	0.4857	0.6840	0.1004
mean	0.4416	0.6162	0.691	0.4828	0.6810	0.1046

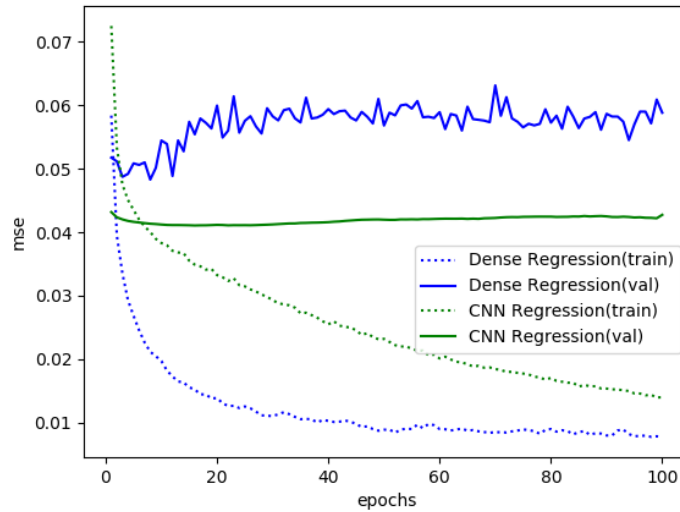


Figure 8.1: The mean training and validation loss of 5-fold cross validation(MSE) for the regression approach.

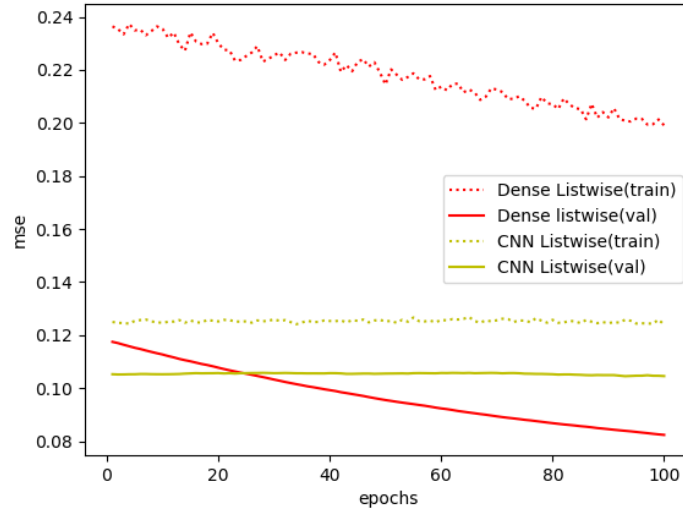


Figure 8.2: The mean training and validation loss of 5-fold cross validation(MSE) for the list-wise approach.

models. In table 8.5 the highest value for each measure for each fold is in bold. For MSE the lowest value is made bold, since this indicates the best result. The last row indicates the mean of the different measures.

When calculating the significance of both models, paired student t-test is used. When comparing on MSE, the difference between the Dense list-wise approach and CNN list-wise approach is significant at 1% with a p-value of $0.0062 < 0.01$. When comparing *KRCC* (p-value is 0.45) and *SRCC* (p-value is 0.3523) no significant difference can be found based on these results.

The training and validation MSE of the regression approaches is plotted in figure 8.1, the MSE during training of the list-wise approach are plotted in figure 8.2.

8.2.3 Results against test set

For the list-wise approach, the Dense network outperformed the CNN model on MSE and the difference is significant(see tables 8.7, 8.8 and 8.9 for the p-values). The results obtained with the easy data can only be compared to results on same data set, or results obtained with the same network. If this is not the case, the paired t-test is not applicable. Therefore the Dense list-wise model is trained on the full data set and tested against the test set. Moreover the CNN regression model outperformed the Dense network outperformed the Dense list-wise model. The CNN regression model is trained on the full data set as well and tested on the test set.

The Dense list-wise model achieved an *KRCC* of 0.3632, an *SRCC* of 0.5151 and an MSE of 0.0723. The CNN regression model achieved an *KRCC* of 0.6049, an *SRCC* of 0.8228 and an MSE of 0.0364.

Table 8.7: The p-values when comparing 5-fold MSE values. Bold if significant at 5%.

	Dense(reg)	CNN(reg)	Dense(list)	CNN(list)	Dense(list, easy)
Dense(reg)		0.1281	0.1332	0.0156	
CNN(reg)			0.0118	0.0002	
Dense(list)				0.0062	
Dense(list, easy)			0.0155		
CNN(reg, easy)		0.1461			0.0004

Table 8.8: The p-values when comparing 5-fold *KRCC* values. Bold if significant at 10%.

	Dense(reg)	CNN(reg)	Dense(list)	CNN(list)	Dense(list, easy)
Dense(reg)		0.2603	0.2786	0.2117	
CNN(reg)			0.0988	0.1015	
Dense(list)				0.4500	
Dense(list, easy)			0.8923		
CNN(reg, easy)		0.2320			0.4243

8.3 Results sensitivity test

To see if the system performance is likely to improve by adding more data, the models are trained on different sized data set. This is done by randomly removing samples in steps of 50. The *KRCC* and *SRCC* can be found in figure 8.3. It can be seen that the Dense list-wise model performance decreases faster than the CNN regression model.

8.4 Remove difficult to rank samples

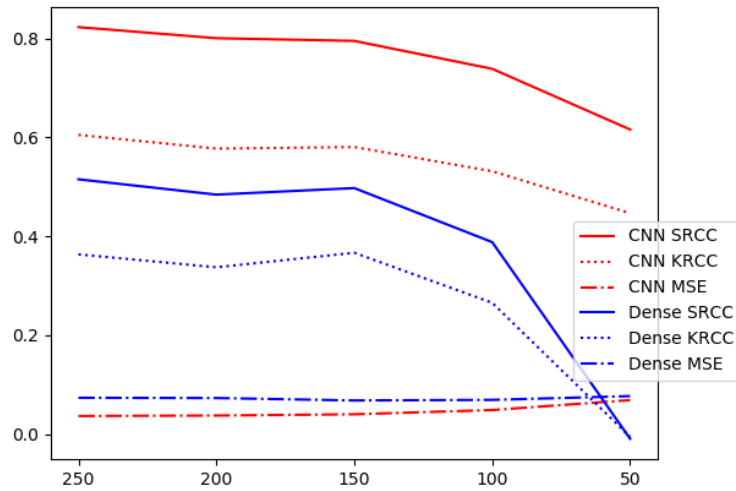
In order to see how the Machine Learning model performs on annotated data on which the annotators agree, a set of 250 samples is selected from the original data. The 50 samples with the highest variance in positions over the different annotators are dropped from the original data. A set of 250 samples is selected so that the results can be compared to the 5-fold cross validation results. This data set is called easy-data. By keeping the training size the same as with the 5-fold cross validation, the results of the models can be compared with the results of the models(with the same architecture) that were trained and tested using 5 fold cross validation. A comparison can be done by applying a paired students t-test. The results of this test can indicate if models perform significantly better if easier data is used.

Tables 8.10 and 8.11 show the *KRCC* and *SRCC* values when comparing the individual annotations of the new data set, yielding a mean *KRCC* of 0.602 and a mean *SRCC* of 0.807.

Table 8.12 shows the in-transitivity values for the individual annotations against the new ground truths, which are generated from the new data set. The linear matrix achieves a

Table 8.9: The p-values when comparing 5-fold *SRCC* values. Bold if significant at 10%.

	Dense(reg)	CNN(reg)	Dense(list)	CNN(list)	Dense(list, easy)
Dense(reg)		0.4333	0.2534	0.2279	
CNN(reg)			0.0892	0.1522	
Dense(list)				0.3523	
Dense(list, easy)			0.8836		
CNN(reg, easy)		0.3237			0.0625

**Figure 8.3:** *KRCC* and *SRCC* for the sensitivity test.

mean in-transitivity value of 0.122, the weighted matrix a mean in-transitivity value of 0.149, both the median matrices a mean in-transitivity value of 0.1223.

Table 8.13 shows the *KRCC* values and *SRCC* values against the new ground truth for each individual annotator, yielding a mean *KRCC* of 0.7558 and an *SRCC* of 0.9235 for the final ranking.

The results of the Dense list-wise approach and the CNN regression approach trained on the easy-data data set are shown in table 8.14. The difference in MSE for the Dense list-wise approaches (comparing easy data with the 5 fold cross validation on the data set) resulted in a significant difference at 5%. The CNN regression model trained on the easy data set outperformed the Dense list-wise model trained on the easy data set and the difference is significant at 1%. The p-values can be found in tables 8.7, 8.8 and 8.9.

Table 8.10: *KRCC* for different Annotators(easy-data).

	Anno. 1	Anno. 2	anno. 3	anno. 4
Annotator 1	1.0	0.619	0.601	0.611
Annotator 2		1.0	0.601	0.587
Annotator 3			1.0	0.593
Annotator 4				1.0

Table 8.11: *SRCC* for different Annotators(easy-data).

	Anno. 1	Anno. 2	anno. 3	anno. 4
Anno. 1	1.0	0.827	0.802	0.815
Anno. 2		1.0	0.808	0.793
Anno. 3			1.0	0.799
Anno. 3				1.0

Table 8.12: In-transitivity versus the different normalized rankings(easy-data).

	Anno. 1	Anno. 2	anno. 3	anno. 4
Linear matrix	0.117	0.125	0.123	0.123
Weighted Matrix	0.146	0.149	0.15	0.151
Median Matrix	0.114	0.123	0.124	0.128
Median Matrix (Eq)	0.114	0.123	0.124	0.128

Table 8.13: *KRCC* and *SRCC* against ground truth(easy-data).

	Anno. 1	Anno. 2	anno. 3	anno. 4
<i>KRCC</i> ranking	0.766	0.75	0.753	0.754
<i>SRCC</i> ranking	0.929	0.924	0.922	0.919

Table 8.14: *KRCC*, *SRCC* and MSE for Dense list-wise and CNN Regression on the same data set using 5-fold cross validation(easy-data). The best results are made bold.

fold n	Dense list-wise			CNN		
	<i>KRCC</i>	<i>SRCC</i>	MSE	<i>KRCC</i>	<i>SRCC</i>	MSE
1	0.3437	0.4966	0.0986	0.5951	0.7917	0.06
2	0.5265	0.7088	0.1059	0.5559	0.7243	0.0683
3	0.5118	0.6776	0.1129	0.6718	0.8597	0.0339
4	0.3355	0.5267	0.1377	0.5429	0.7444	0.0778
5	0.5396	0.7345	0.1124	0.5249	0.7467	0.0587
mean	0.4514	0.6286	0.1135	0.5781	0.7733	0.0597

Conclusion

The following conclusions are based on the results presented in chapters 6, 7 and 8.

9.1 RQ1: How can 3000 files be efficiently annotated?

9.1.1 SQ1.1: What is the level of transitivity in aggression annotation in audio samples?

To answer this question the pairwise comparison in the first experiment is done. The level of transitivity is $1 - \text{in-transitivity}$, which is 0.85 where perfect transitivity would be 1. When working with real world data in-transitivity can be expected, as state in the background(2.8.1). Since 0.85 is a reasonably high value, this indicates that the annotators agree globally on the order of audio samples when ranking on aggression. This means that humans can order the samples on aggression, which makes it interesting to see if machine learning can be applied to automatically determine this order.

9.1.2 How can audio samples be annotated into a ranking based on aggression?

In this project pairwise rank annotation is compared to list-wise rank annotation. Both methods are used in early research(such as [10]) where participants had to rank audio samples. In earlier research the methods were used on smaller data sets, for scalability the binary comparison and multi-list approach are introduced. In the first experiment four methods are compared: the pairwise comparison method, in which all samples are compared with each other; the binary comparison approach, in which a ranking is build using the binary insertion sort algorithm; the list-wise approach, where all samples are compared in one list; and the multi-list approach, in which two list containing 5 samples plus two anchors are compared.

9.1.3 How efficient are the different types of audio annotation when ranking based on aggression?

The efficiency of all individual systems can be found in chapter 6. The most efficient method tested in this research is binary comparison. The results of the first experiment show that this method yields more transitive results than the list-wise and multi-list approach. Even the baseline approach, in which all samples are pairwise compared, is beaten by the binary comparison method.

The first experiment also shows that the binary comparison method is most time efficient. The values obtained through the sliders in the list-wise and multi-list approaches carry too little information to make these approaches time efficient. Since participants would always distribute the sliders uniformly, the value difference between sample a and sample b and the value difference between sample a and sample c does not hold information about the difference between b and c if these samples were in different windows. That is if both b and c are ranked as more, or both are ranked as less aggressive than a . These results are in line with previous research such as [7]–[10] This makes the list-wise approaches inefficient, because only one new sample can be added at a time. The fact that the list-wise approaches narrow the search field smaller than the binary approach after every iteration does not way out the time loss per iteration.

Annotation Results

Binary Comparison is used to annotate a set of 300 audio samples. The data is annotated by 4 people, the results can be found in chapter 7. The rankings of the different achieved moderate strong correlation on both $KRCC$ and $SRCC$. This indicates that the different annotators agreed moderate to strong with each other on the order of audio samples. Furthermore the in-transitivity value of the different annotators compared to four different groundtruth matrices is comparable to that of the baseline in the first experiment, with a mean in-transitivity value of 0.151 for the linear matrix, and 0.149 for the median matrix. The median matrix reaches the lowest in-transitivity value. Median aggregation proved the best method in data simulation as well. Therefore this is used to build the final ranking. This final ranking yields even stronger correlation on both $KRCC$ and $SRCC$. The chosen annotation method yielded similar rankings over the different annotators; this means it is a useful method to annotate audio samples based on aggression.

9.2 How can audio samples automatically be ranked based on aggression?

9.2.1 How can regression be applied to rank audio files based on aggression?

Two models are developed in this project. A fully connect network, referred to as Dense, and a convolutional neural network, referred to as CNN. The result of the 5-fold cross validation as well as the results on the test set are shown in chapter 8.

Both models performed comparable on *KRCC* and *SRCC*. Dense achieved a mean *KRCC* 0.5109 and *SRCC* of 0.7111, the CNN achieved a *KRCC* of 0.536 and *SRCC* of 0.7363 on the 5-fold cross validation. Furthermore no statistical significant difference was found in the results of the 5-fold cross validation. Both methods achieved lower values than the annotators(*KRCC*: 0.698, *SRCC*: 0.88). Still the results of the models both indicate moderate to strong positive correlation. Since the CNN performed best, this model is chosen as the regression model that is tested on the test set, on which it achieved an *KRCC* of 0.6049 and an *SRCC* of 0.8228, indicating moderate to strong correlation. The moderate to strong correlations indicate that a regression model can be used to predict a linear target, which can be used to order the samples based on aggression.

9.2.2 How can list-wise learning-to-rank be applied to rank audio files based on aggression?

Both list-wise models are based on the same architectures as the regression models. The models are not equal in terms of hyper parameters, because they are both optimized using Bayesian optimization with a loss based on the output of that model. Both methods perform similar and no statistically significant difference can be found based on these results. The Dense list-wise model achieved a mean *KRCC* of 0.4416 and a mean *SRCC* of 0.6162 indicating moderate correlation. The CNN list-wise model achieved a mean *KRCC* of 0.4333 and a mean *SRCC* of 0.6810 indicating moderate to strong correlation. Based on MSE significant difference is found, indicating that the Dense list-wise model(0.0691) outperforms the CNN list-wise model(0.1046). Since this is the only significant difference, the Dense list-wise model is chosen as best model and tested on the test set, achieving an *KRCC* of 0.3632, an *SRCC* of 0.5151 and an MSE of 0.0723. These results indicate a weak to moderate correlation between the ground truth of the test set and the predicted ranking.

9.2.3 List-wise versus Regression

When comparing the rank coefficients, only in one combination a statistical significant difference was found. The CNN Regression model outperforms the Dense list-wise model. When comparing MSE the CNN list-wise model is outperformed by the Dense list-wise model as

well as the Dense Regression model. The CNN regression model achieved an MSE of 0.0364.

When comparing the best list-wise model and the best regression model, the regression model outperformed the list-wise model. Based on the results in this paper, the regression approach is better suitable for aggression based audio ranking.

9.2.4 Sensitivity test

The sensitivity test yielded stable results. In figure 8.3 it can be seen that the *KRCC* and *SRCC* greatly improve until 150 training samples. From 150 onward the results are stable. Therefore, the sensitivity test did not give any indication of data shortage. In machine learning problems, more data leads to better results in most cases, but this conclusion cannot be drawn from the results in this report.

9.2.5 Easy data

It is expected that the annotation scores improve, if the samples that cause errors are removed. On the other hand, the scores of automatic models did not change a lot: Dense list-wise model trained on the standard data set achieved a *KRCC* of 0.4416 and a *SRCC* of 0.6162, on the easy-data data set this system achieved a *KRCC* of 0.4514 and a *SRCC* of 0.6286. Both rank measures slightly improved however the difference is not significant. The Dense list-wise model achieved an MSE of 0.0691, which increased to 0.1135 when training on the easy data. This difference is in fact significant and might be due to over-training.

The CNN Regression achieved a *KRCC* of 0.5360 and a *SRCC* of 0.7363 on the standard data set and a *KRCC* of 0.5781 and a *SRCC* of 0.7733 on the easy-data data set, showing slight decrease on both measures however not significant. The MSE for the CNN regression model increased from 0.0427 to 0.0597 when changing to the easy data, however this difference is not significant.

Machine Learning algorithms aim to learn two things from data. First of all, these algorithms try to fit the data. This means extracting relevant features in order to predict a target belonging to a value. Next to that, these algorithms try to recognize noise in the data in order to ignore this noise. By removing difficult to rank samples, you are essentially removing noise. This can be an explanation for the fact that removing difficult samples did not improve the performance of the models, because the initial models were already capable to distinguish noise. Something in these recordings made it difficult for the annotators to rank these samples. This might be due to noise which the models had already learned and did not influence the ranking.

Discussion

In this section the conclusions(chapter 9) of this report are discussed. First the final performance is compared to related research. This is followed by some notes on the evaluation measures. Finally some future research is suggested.

10.1 Results Compared to Kooij et al.

Kooij et al. [48], as discussed in chapter 3, build an aggression detection system, including audio classification. They measure the performance of the system in Root Mean Squared Error(RMSE) and achieved an RMSE of 0.223 for the audio classification system and an RMSE of 0.188 when using the full system. The CNN regression model achieved an MSE of 0.0364, to compare this with the results of Kooij et al. the square root of this value is calculated, which translate to a value of 0.1907. The performance of the models in this paper is in the same range as the results found by Kooij et al. [48].

10.1.1 Data annotation very time consuming

The annotation of the data using the binary comparison method is very time consuming. Some samples are very similar while others are so different that it is difficult to compare them. It might be interesting to see if it is easier to compare only samples from the same source. In this project audio samples from different sources are used. It is for example very difficult to compare a sample recorded in a correctional institution with a sample recorded in a health care institution. This has multiple reasons. The recording sounds different due to different recording hardware and room acoustics. Another reason is the interpretation of aggression. This interpretation depends on the place where an event occurs. Something might be judged as aggressive in a health care institution, but when the same even occurs in a correctional institution it might not be judged as aggressive. This might make the current approach generalize better over different situations, but this might go at a cost of precision in individual situations.

10.2 *SRCC* vs *KRCC*

In every case in this paper, *SRCC* is higher than *KRCC*. This goes for the annotation part as well as the machine learning part. A probable cause of this is explained through the following example:

$$a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$$

$$b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]$$

$$d = [5, 1, 2, 3, 4, 0, 6, 7, 8, 9]$$

$$SRCC(a, b) = 0.4545$$

$$KRCC(a, b) = 0.6$$

$$SRCC(a, c) = 0.6969$$

$$KRCC(a, c) = 0.6$$

The example above shows that *SRCC* penalizes a big positional difference harder than *KRCC*. So if one sample is far out of place, *SRCC* will drop more than *KRCC*.

$$d = [1, 0, 3, 2, 5, 4, 7, 6, 9, 8]$$

$$SRCC(a, c) = 0.9394$$

$$KRCC(a, c) = 0.7778$$

In above example, all samples are switched one place. No sample is in its original position, but the samples are only one position out of place. This kind of error is penalized harder by *KRCC* than by *SRCC*.

It seems that the latter error occurs throughout this project, samples are switched but not far out of place. This explains that *KRCC* is lower than *SRCC* in every case in this paper.

10.3 Automatic Ranking of Audio Based on Aggression

This paper proves that audio samples can be ordered based on aggression by annotators. Next to that the paper proves that audio can automatically be ranked based on aggression. The automatic scores might improve by applying different machine learning techniques such as Recurrent Neural Networks and other combination of networks. These techniques are explained in the background(2.3) and State of the Art(3). Recurrent Neural Networks are widely used in emotion detection and audio processing. The research shows great improvements in system performance when applying these techniques.

10.4 Value based treshold

Although the list-wise approaches seem to be able to learn an aggression ranking in audio samples, this approach does not optimize the target value for the individual samples very well. This is indicated by the low MSE achieved by both list-wise models. The goal of Sound Intelligence is to build a sensitivity control. It is useful if this sensitivity control can be set by a value between 0 and 1. For this a low MSE is needed. Next to that, the regression model outperformed the list-wise model on the ranking task. Therefore it is more useful for Sound Intelligence to use a regression based algorithm.

10.5 Future Research

10.5.1 Different Machine Learning Techniques

As described in the background(2.3) and state of the art(3), different machine learning techniques are used in audio processing and affective computing. Techniques such as Recurrent Neural Networks show great improvements in this field. It would be interesting to see if applying these techniques could improve the performance on the data set defined in this paper.

10.5.2 Data Source

The feedback of the annotators(7.5 indicated that the combination of different data sources makes it difficult to compare samples. It would be interesting to see if annotators deliver more similar rankings when only samples from one source are used. The data set recorded in the correctional facility would be an interesting data set for this purpose, because this data set covers the full aggression scale. Next to the annotation results, it would be interesting to see if machine learning models are better able to rank such a data set.

10.5.3 Exclude Irrelevant Samples

The annotators indicated(7.5 that some samples were irrelevant when it comes to aggression. These samples are still being ranked into this data set. It would be interesting to see if the results would improve if annotators could exclude certain samples.

Acknowledgement

I would like to thank Sound Intelligence for the opportunity and resources to do this research project. During this project Jasper van Dorp Schuitman has been a great supervisor, for which I am grateful. Next to that, I want to thank Maarten Duijndam, for giving me the opportunity.

In addition, I would like to thank Mannes Poel for his guidance during the graduation project as well as the preliminary research topics. Furthermore, I would like to thank Khiet Truong for her feedback during the project.

Finally, I want to thank the Sound Intelligence annotation team for their time. I know the ranking of the samples was a tedious and time consuming job.

Bibliography

- [1] T. S. Gunawan, M. F. Alghifari, M. A. Morshidi, and M. Kartiwi, "A review on emotion recognition algorithms using speech analysis," *Indonesian Journal of Electrical Engineering and Informatics (IJEI)*, vol. 6, no. 1, pp. 12–20, 2018.
- [2] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol. 7, pp. 19 143–19 165, 2019.
- [3] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.
- [4] P. Susini, G. Lemaitre, and S. McAdams, "Psychological measurement for sound description and evaluation," *Measurements with persons: Theory, methods, and implementation areas*, vol. 227, 2012.
- [5] A. M. Colman, *A dictionary of psychology*. Oxford University Press, USA, 2015.
- [6] J. A. Russell, "A circumplex model of affect," *Journal of personality and social psychology*, vol. 39, no. 6, p. 1161, 1980.
- [7] P. Chevret and E. Parizet, "An efficient alternative to the paired comparison method for the subjective evaluation of a large set of sounds," in *19th International Congress on Acoustics*, 2007.
- [8] E. Parizet, E. Guyader, and V. Nosulenko, "Analysis of car door closing sound quality," *Applied acoustics*, vol. 69, no. 1, pp. 12–22, 2008.
- [9] E. Poirson, J.-F. Petiot, and F. Richard, "A method for perceptual evaluation of products by naive subjects: Application to car engine sounds," *International Journal of Industrial Ergonomics*, vol. 40, no. 5, pp. 504–516, 2010.
- [10] E. Parizet, N. Hamzaoui, and G. Sabatie, "Comparison of some listening test methods: a case study," *Acta Acustica united with Acustica*, vol. 91, no. 2, pp. 356–364, 2005.
- [11] J. G. Ackenhusen, *Real-Time Signal Processing: Design and Implementation of Signal Processing Systems*. Prentice Hall PTR, 2007.

- [12] M. Sahidullah and G. Saha, "Design, analysis and experimental evaluation of block based transformation in mfcc computation for speaker recognition," *Speech Communication*, vol. 54, no. 4, pp. 543–565, 2012.
- [13] V. Tyagi and C. Wellekens, "On desensitizing the mel-cepstrum to spurious spectral components for robust speech recognition," in *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, vol. 1. IEEE, 2005, pp. I–529.
- [14] V. Tiwari, "Mfcc and its applications in speaker recognition," *International journal on emerging technologies*, vol. 1, no. 1, pp. 19–22, 2010.
- [15] M. El Ayadi, M. S. Kamel, and F. Karray, "Survey on speech emotion recognition: Features, classification schemes, and databases," *Pattern Recognition*, vol. 44, no. 3, pp. 572–587, 2011.
- [16] W. Dai, C. Dai, S. Qu, J. Li, and S. Das, "Very deep convolutional neural networks for raw waveforms," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 421–425.
- [17] W. S. Noble, "What is a support vector machine?" *Nature biotechnology*, vol. 24, no. 12, p. 1565, 2006.
- [18] K. Gurney, *An introduction to neural networks*. CRC press, 2014.
- [19] S. Skansi, *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Springer, 2018.
- [20] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:1402.1128*, 2014.
- [21] Z. Wu and S. King, "Investigating gated recurrent networks for speech synthesis," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 5140–5144.
- [22] N. D. Lane, P. Georgiev, and L. Qendro, "Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2015, pp. 283–294.
- [23] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *Machine Learning for Signal Processing (MLSP), 2015 IEEE 25th International Workshop on*. IEEE, 2015, pp. 1–6.
- [24] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2392–2396.

- [25] Q. Kong, Y. Xu, W. Wang, and M. D. Plumbley, "A joint detection-classification model for audio tagging of weakly labelled data," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 641–645.
- [26] L. Hertel, H. Phan, and A. Mertins, "Classifying variable-length audio files with all-convolutional networks and masked global pooling," *arXiv preprint arXiv:1607.02857*, 2016.
- [27] H. Phan, L. Hertel, M. Maass, and A. Mertins, "Robust audio event recognition with 1-max pooling convolutional neural networks," *arXiv preprint arXiv:1604.06338*, 2016.
- [28] J. Jiao, "A framework for finding and summarizing product defects, and ranking helpful threads from online customer forums through machine learning," Ph.D. dissertation, Virginia Tech, 2013.
- [29] P. Li, Q. Wu, and C. J. Burges, "Mcrank: Learning to rank using multiple classification and gradient boosting," in *Advances in neural information processing systems*, 2008, pp. 897–904.
- [30] N. Usunier, D. Buffoni, and P. Gallinari, "Ranking with ordered weighted pairwise classification," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 1057–1064.
- [31] R. Lotfian and C. Busso, "Retrieving categorical emotions using a probabilistic framework to define preference learning samples." in *INTERSPEECH*, 2016, pp. 490–494.
- [32] C. Fang, H. Zhang, M. Zhang, and J. Wang, "Recommendations based on listwise learning-to-rank by incorporating social information." *KSII Transactions on Internet & Information Systems*, vol. 12, no. 1, 2018.
- [33] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
- [34] S. Jachner, G. Van den Boogaart, T. Petzoldt *et al.*, "Statistical methods for the qualitative assessment of dynamic models with time delay (r package qualv)," *Journal of Statistical Software*, vol. 22, no. 8, pp. 1–30, 2007.
- [35] P. J. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, pp. 73–101, 03 1964.
- [36] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, "Listwise approach to learning to rank: theory and algorithm," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1192–1199.
- [37] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in science conference*. Citeseer, 2013, pp. 13–20.

- [38] R. Fagin, R. Kumar, and D. Sivakumar, "Comparing top k lists," *SIAM Journal on discrete mathematics*, vol. 17, no. 1, pp. 134–160, 2003.
- [39] H. Abdi, "The kendall rank correlation coefficient," *Encyclopedia of Measurement and Statistics*. Sage, Thousand Oaks, CA, pp. 508–510, 2007.
- [40] M. M. Mukaka, "A guide to appropriate use of correlation coefficient in medical research," *Malawi Medical Journal*, vol. 24, no. 3, pp. 69–71, 2012.
- [41] H. Akoglu, "User's guide to correlation coefficients," *Turkish journal of emergency medicine*, 2018.
- [42] M. Regenwetter, J. Dana, and C. P. Davis-Stober, "Transitivity of preferences." *Psychological review*, vol. 118, no. 1, p. 42, 2011.
- [43] A. Tversky, "Intransitivity of preferences." *Psychological review*, vol. 76, no. 1, p. 31, 1969.
- [44] L. S. Temkin, "A continuum argument for intransitivity," *Philosophy & Public Affairs*, vol. 25, no. 3, pp. 175–210, 1996.
- [45] A. Rubinstein, "Similarity and decision-making under risk (is there a utility theory resolution to the allais paradox?)," *Journal of economic theory*, vol. 46, no. 1, pp. 145–153, 1988.
- [46] D. Harman, "Information retrieval evaluation," *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 3, no. 2, pp. 1–119, 2011.
- [47] Y. Wang, L. Wang, Y. Li, D. He, W. Chen, and T.-Y. Liu, "A theoretical analysis of ndcg ranking measures," in *Proceedings of the 26th annual conference on learning theory (COLT 2013)*, vol. 8, 2013, p. 6.
- [48] J. F. Kooij, M. Liem, J. D. Krijnders, T. C. Andringa, and D. M. Gavrilu, "Multi-modal human aggression detection," *Computer Vision and Image Understanding*, vol. 144, pp. 106–120, 2016.
- [49] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck, and D. Turnbull, "Music emotion recognition: A state of the art review," in *Proc. ISMIR*, vol. 86. Citeseer, 2010, pp. 937–952.
- [50] R. Lotfian and C. Busso, "Practical considerations on the use of preference learning for ranking emotional speech," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5205–5209.
- [51] G. N. Yannakakis and H. P. Martinez, "Grounding truth via ordinal annotation," in *2015 international conference on affective computing and intelligent interaction (ACII)*. IEEE, 2015, pp. 574–580.

- [52] Y.-H. Yang and H. H. Chen, "Music emotion ranking," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2009, pp. 1657–1660.
- [53] —, "Ranking-based emotion recognition for music organization and retrieval," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 762–774, 2011.
- [54] H. P. Martinez, G. N. Yannakakis, and J. Hallam, "Dont classify ratings of affect; rank them!" *IEEE transactions on affective computing*, vol. 5, no. 3, pp. 314–326, 2014.
- [55] A. Metallinou and S. Narayanan, "Annotation and processing of continuous emotional attributes: Challenges and opportunities," in *2013 10th IEEE international conference and workshops on automatic face and gesture recognition (FG)*. IEEE, 2013, pp. 1–8.
- [56] G. N. Yannakakis and J. Hallam, "Ranking vs. preference: a comparative study of self-reporting," in *International Conference on Affective Computing and Intelligent Interaction*. Springer, 2011, pp. 437–446.
- [57] G. N. Yannakakis and H. P. Martínez, "Ratings are overrated!" *Frontiers in ICT*, vol. 2, p. 13, 2015.
- [58] S. Tognetti, M. Garbarino, A. Bonarini, and M. Matteucci, "Modeling enjoyment preference from physiological responses in a car racing game," in *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*. IEEE, 2010, pp. 321–328.
- [59] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.
- [60] S. Parthasarathy, R. Lottian, and C. Busso, "Ranking emotional attributes with deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 4995–4999.
- [61] L.-C. Yang, S.-Y. Chou, J.-Y. Liu, Y.-H. Yang, and Y.-A. Chen, "Revisiting the problem of audio-based hit song prediction using convolutional neural networks," *arXiv preprint arXiv:1704.01280*, 2017.
- [62] H. Li, "Learning to rank for information retrieval and natural language processing," *Synthesis Lectures on Human Language Technologies*, vol. 7, no. 3, pp. 1–121, 2014.
- [63] E. Ghanbari and A. Shakery, "Err. rank: An algorithm based on learning to rank for direct optimization of expected reciprocal rank," *Applied Intelligence*, vol. 49, no. 3, pp. 1185–1199, 2019.
- [64] P. Diehr, D. C. Martin, T. Koepsell, and A. Cheadle, "Breaking the matches in a paired t-test for community interventions when the number of pairs is small," *Statistics in medicine*, vol. 14, no. 13, pp. 1491–1504, 1995.

Statistical Formulas

A.1 Z-scores

Z-scores are used as a normalization of experimental results around the sample mean. The z-score of a sample defines the distance of that sample to the mean in terms of standard deviation. The z-score is calculated using the following formula:

$$z = \frac{x - \mu}{\sigma}$$

where μ = Sample mean,
 σ = Standard deviation

A.2 Paired Students T-Test

Paired students T-Test is used to compare two means, where under the null hypothesis equal means are assumed. The paired t-test compares samples drawn from the same group, as oppose to independent sample T-Test, which compares two independent sample means. To use paired students T-Test, the t value has to be calculated as follows [64]:

$$t = \frac{\bar{d}}{\frac{S_d}{\sqrt{N}}}$$

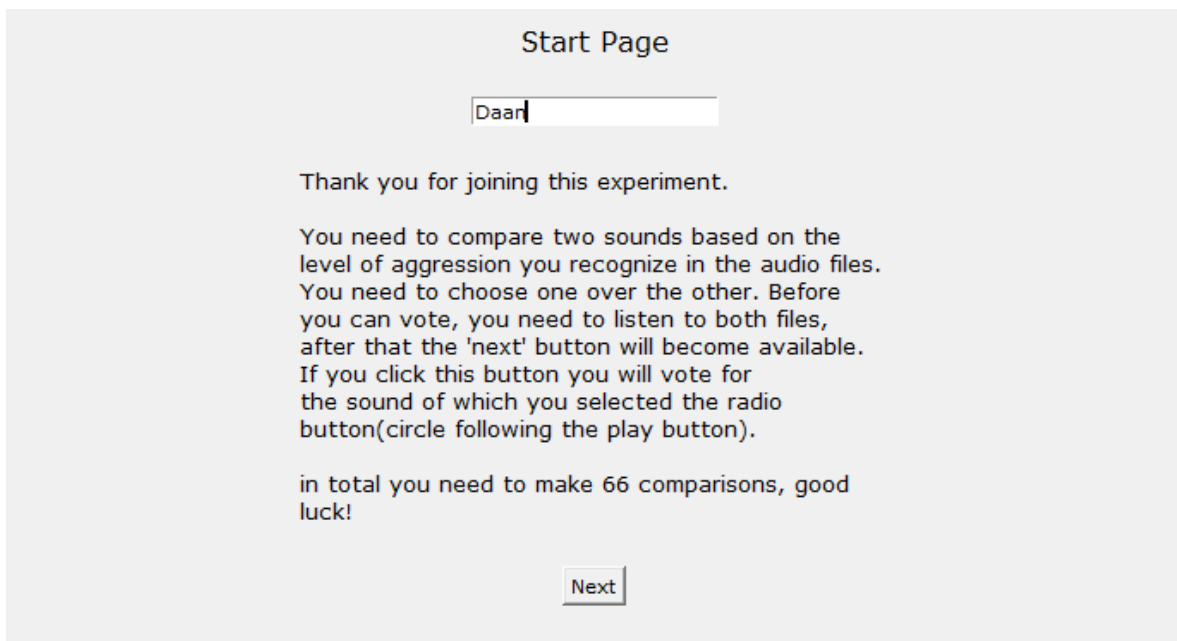
where \bar{d} = Mean difference,
 S_d = Standard deviation,
 N = Sample size

The t value has to be compared to with the t table with $N - 1$ degrees of freedom, resulting in a p-value which can be interpreted as the probability of the difference occurring by chance. This means that if the p-value is high, equal means can be assumed. If the p-value is low, different means can be assumed.

Pairwise Comparison Experiment Tool

The Pairwise tool (figure B.1) is used to compare every sound with every other sound. The results of this test are used as a baseline ranking. The tool loads and shuffles a set of 12 sounds. After that it makes a set containing every distinct pair of samples in the set of sounds, where $x, y = y, x \forall x, y$. The set of pairs is then again shuffled.

The participant has to compare all pairs based on aggression. Before the participant can vote for the most aggressive sample, both sounds in the pair have to be played at least once (figure B.2). After all pairs are compared, the votes, total time spent, time spent per comparison and times played per sound are saved.



The screenshot shows the 'Start Page' of the experiment tool. At the top, the title 'Start Page' is centered. Below it is a text input field containing the name 'Daan'. The main body of the page contains a series of instructions: 'Thank you for joining this experiment.', 'You need to compare two sounds based on the level of aggression you recognize in the audio files. You need to choose one over the other. Before you can vote, you need to listen to both files, after that the 'next' button will become available. If you click this button you will vote for the sound of which you selected the radio button(circle following the play button).', and 'in total you need to make 66 comparisons, good luck!'. At the bottom center, there is a 'Next' button.

Figure B.1: Start screen of the paired comparison tool

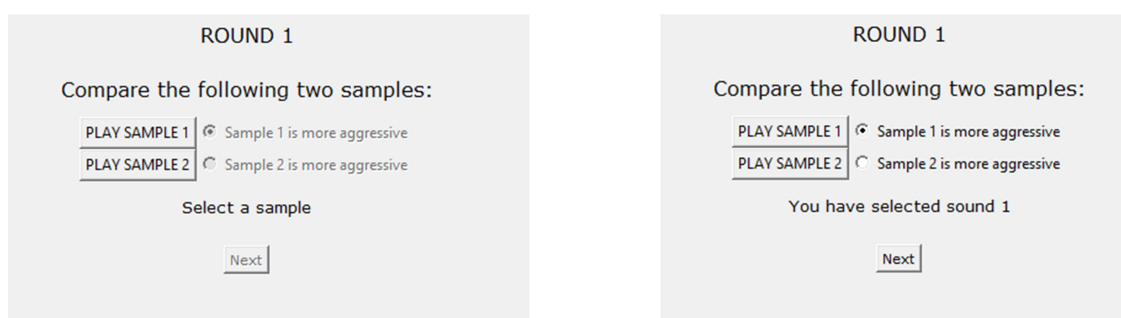
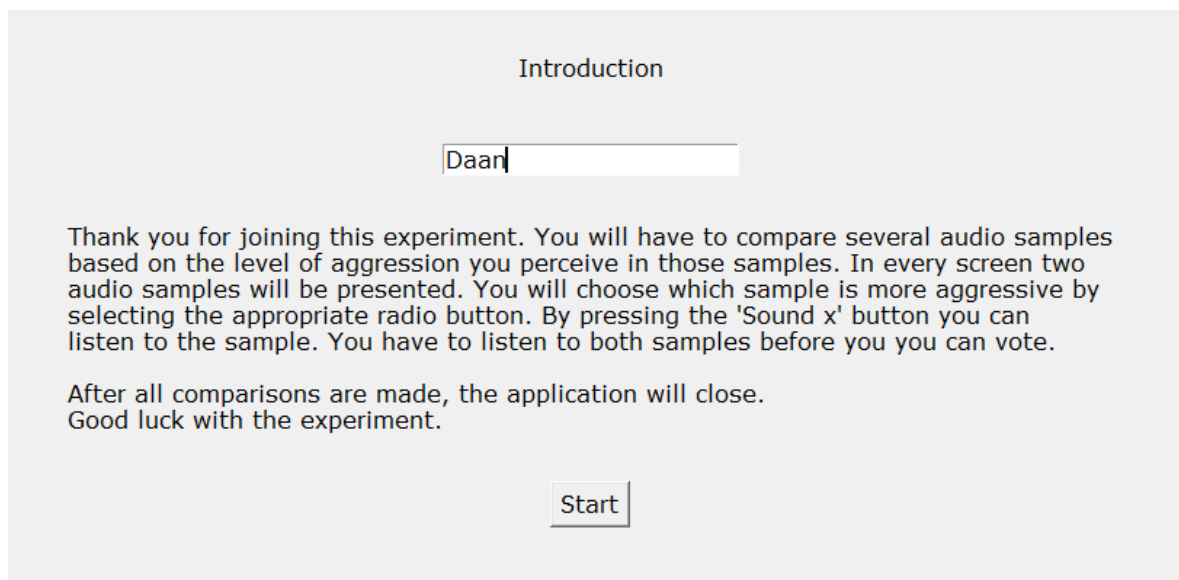


Figure B.2: Comparison screen before and after both sounds are played

Binary Comparison Experiment Tool

The Binary tool (figure C.1) first loads and shuffles the 12 audio samples. The first sample in this shuffled list is used as initial ranked list. The other samples are added by comparing them individually to the ranked list. First a sample is compared to the middle sample in the ranked list. If this is the last sample to compare to, the new sample is added before or after this sample in the ranked list, depending on the outcome of the comparison. If this is not the last sample to compare to, the compare window shifts. If the new sample is evaluated less aggressive, the middle sample-1 of the compare window becomes the upper bound of the new compare window. If the sample is evaluated more aggressive, the middle sample +1 becomes the upper bound of the new compare window. This method is repeated until all samples are in the ranked list.

Before the participant can vote, both samples of the comparison have to be played. The screens look similar to the screens of the paired comparison tool.



Introduction

Daan

Thank you for joining this experiment. You will have to compare several audio samples based on the level of aggression you perceive in those samples. In every screen two audio samples will be presented. You will choose which sample is more aggressive by selecting the appropriate radio button. By pressing the 'Sound x' button you can listen to the sample. You have to listen to both samples before you you can vote.

After all comparisons are made, the application will close.
Good luck with the experiment.

Start


Figure C.1: Start screen of the binary compassion tool

List-wise Comparison Experiment Tool

The list-wise tool (figure D.1) loads and shuffles the 12 audio samples. On the screen a play button and a slider is shown for every sample. The sliders are initialized in the middle. The participant can raise the slider if he evaluates the belonging sample as aggressive and lower the slider if he evaluates the sample as not aggressive. When all individual samples are played and evaluated the samples can be sorted, yielding a ranking from least to most aggressive. The participant can now evaluate this ranking and save it if he agrees.

Below you see 12 sliders and 12 play buttons. The slider belongs to the play button below it. It is your job to evaluate the 12 sounds based on the level of aggression you recognize in the samples. If you perceive a sample as aggressive, move the scale up. If you perceive the sample to be not aggressive at all, move the scale down. It is advisable to first listen to the sound and evaluate on the slider scale. After that you can press the sort button to sort the samples in Ascending order. After that you can fine tune the results and sort again.

Very Aggressive
Not Aggressive at all




sound 1sound 2sound 3sound 4sound 5sound 6sound 7sound 8sound 9sound 10sound 11sound 12

You still have to play: [4, 5, 6, 7, 8, 9, 10, 11, 12]

Below you see 12 sliders and 12 play buttons. The slider belongs to the play button below it. It is your job to evaluate the 12 sounds based on the level of aggression you recognize in the samples. If you perceive a sample as aggressive, move the scale up. If you perceive the sample to be not aggressive at all, move the scale down. It is advisable to first listen to the sound and evaluate on the slider scale. After that you can press the sort button to sort the samples in Ascending order. After that you can fine tune the results and sort again.

Very Aggressive
Not Aggressive at all



sound 2sound 4sound 10sound 12sound 6sound 8sound 7sound 1sound 11sound 3sound 9sound 5

You still have to play: [4, 5, 6, 7, 8, 9, 10, 11, 12]

Figure D.1: Multi-list comparison

List-wise Comparison Experiment Tool

The Multi-list tool (figure E.1) works very similar to the list-wise tool, but now two lists have to be evaluated. The lists both consist of seven samples. First the Multi-list tool loads the 12 samples, it extracts the least and most aggressive sample (obtained through the Pairwise comparison experiment). It shuffles the other samples and splits it in two lists, the least and most aggressive samples are shuffled through the two lists. The two lists have to be evaluated in the same manner as the list-wise sample. A final ranking is build based on the values of the sliders.

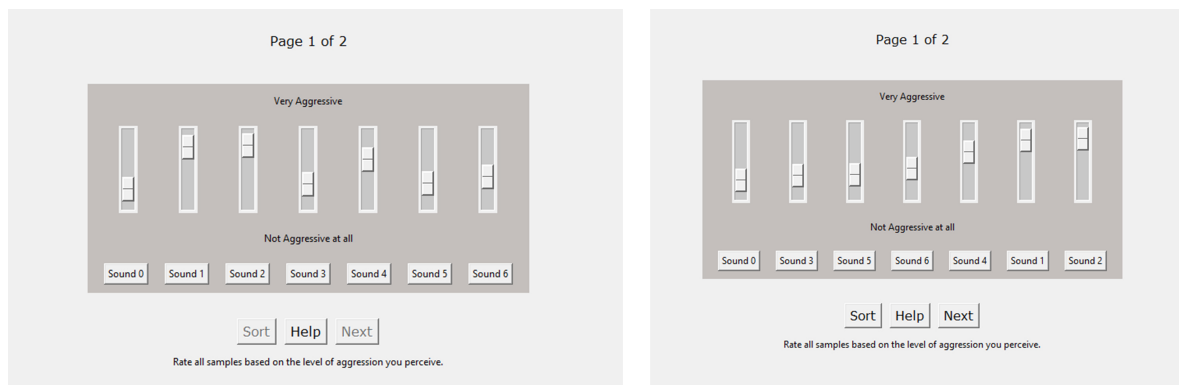


Figure E.1: Multi-list comparison

Annotation Tool

The annotation tool (figure F.1) is based on the binary comparison technique. The tool is functionally the same as the binary comparison tool, but runs on a Sound Intelligence server so that annotators can use it remotely.



Figure F.1: Annotation tool user interface

Appendix G

Simulation of Annotation Process

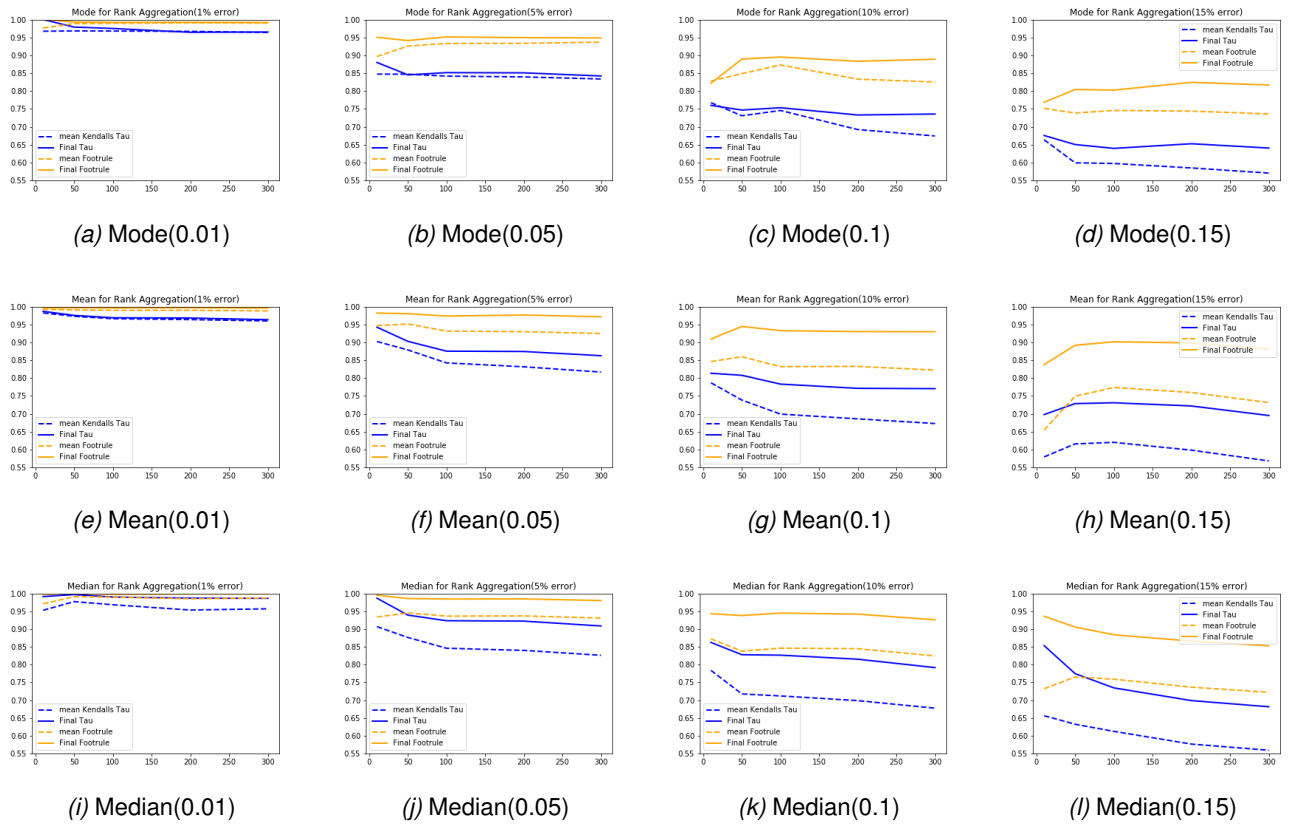


Figure G.1: 3 annotators

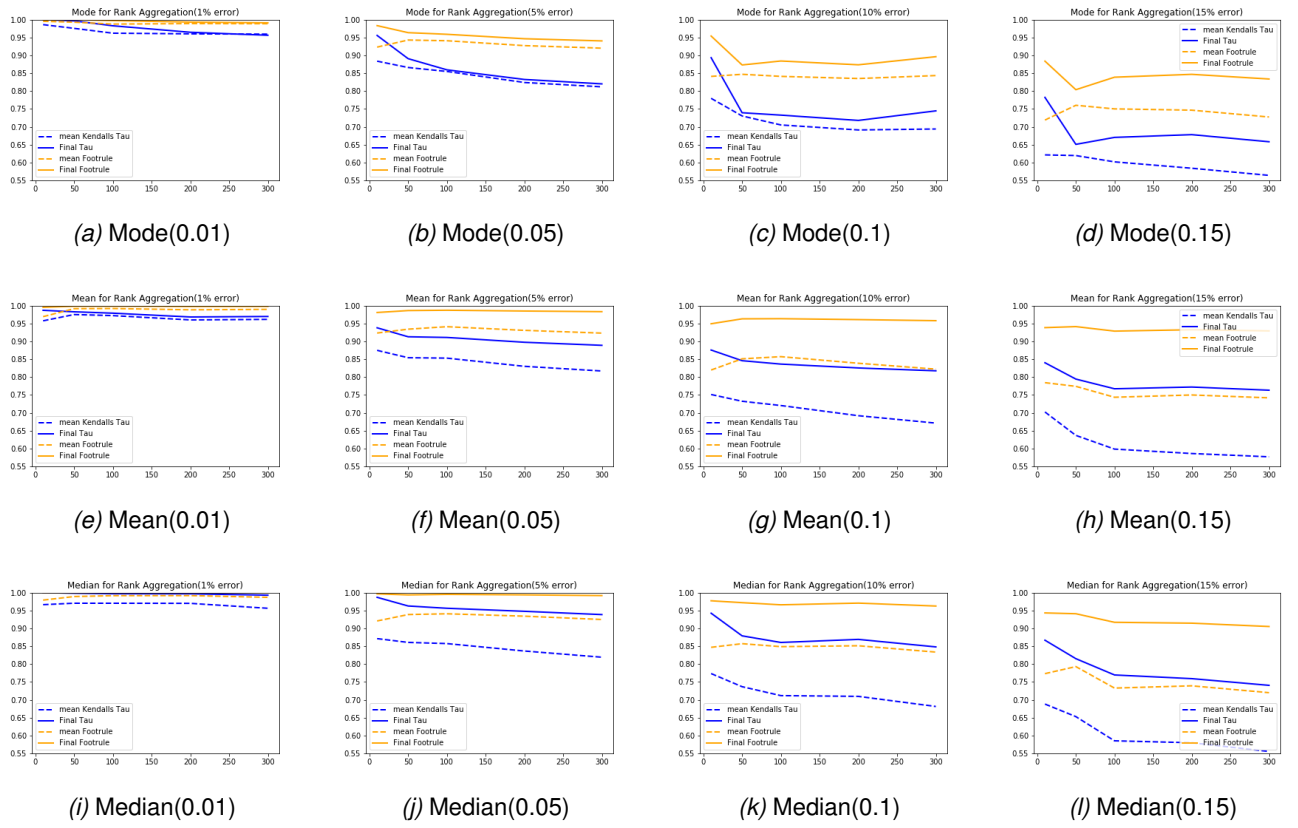


Figure G.2: 5 annotators

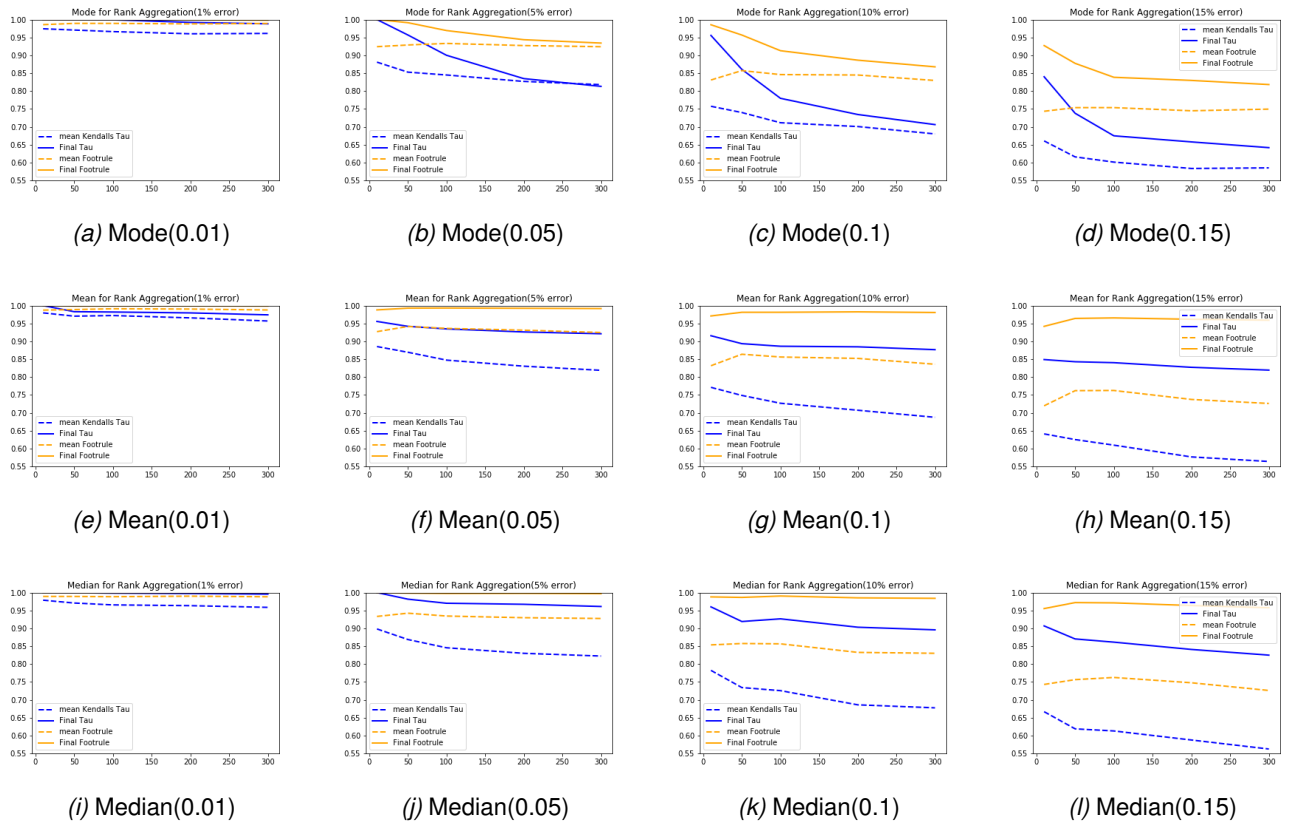


Figure G.3: 10 annotators