A Machine Learning Model for Laughter Detection

Lasse Wolter

4th Year Project Report
Artificial Intelligence and Computer Science
School of Informatics
University of Edinburgh

2022

Abstract

This project investigates machine learning models for laughter recognition, with a focus on integrating automatic laughter detection into a video call system to provide a more engaging speaker experience. Following evaluation of an existing, pre-trained laughter recognition model on the ICSI corpus, I retrain the model on the ICSI corpus and investigate the impact of varying training data on performance. The performance across all models is poor, so I investigate the raw data of the ICSI corpus. I find that non-transcribed regions often contain cross-talk from other individuals, disproving my initial assumption of silence in such regions. This insight is vital for the continuation of this project. The machine learning pipeline for training and evaluating a laughter detection model on the ICSI corpus was published for future work Wolter (2022a).

Acknowledgements

Thank you, God, for always loving me - even if I neglect you.

All glory goes to you.

Table of Contents

Chapter 1

Introduction

The initial title of the project was *A Zoom filter for applause and laughter*. The idea of this project was to create a third option for video-meeting participants. In addition to `mute` and `unmute`, a new option called `filtered` should be added. In `filtered`-mode, a user is automatically unmuted when laughter or applause is detected. The motivation behind this feature is a more engaging speaker experience in virtual conferences. The project's supervisor is regularly involved in such conferences and wishes that reactions like laughter and applause were audible to him.

To achieve the original goal, a real-time machine learning model that detects laughter and applause is required. This project narrows the scope to a machine learning model for laughter detection. Real-time constraints are considered theoretically (Section 6.1) but adapting the model implementation for real-time usage is left for future work. The project was renamed to *A machine learning model for laughter detection* as the focus of the project was set on laughter detection; the beginning of the thesis briefly covers some aspects of applause detection.

Due to poor results of the final model, this thesis outlines the process I have taken, issues I faced and states my final hypothesis about the results:

"A significant number of non-transcribed regions in the ICSI corpus are not actually silence; they are quieter recordings of speech by other participants. Thus, the assumption that non-transcribed regions and laughter can be easily distinguished is wrong."

The main achievements of this project are:

1. Evaluation of Gillick et al.'s (2021) model on the whole ICSI corpus (Chapter 3)

2. Creation of a machine learning pipeline for training and evaluating laughter recognition on the ICSI meeting corpus (Chapter 4)
    - This code is publicly available for future work (Wolter, 2022a)
3. Investigation of training data on model performance (Chapter 5)
    - Includes inspection of raw audio data that led to my final hypothesis

The thesis starts off with a review of existing approaches about laughter and applause detection (Chapter 2). It introduces some evaluation metrics used and gives a more detailed view of the research by Gillick et al. (2021) which is the basis of my work. In Chapter 3 the best model from Gillick et al. (2021), trained on the Switchboard corpus (Godfrey et al., 1992), is evaluated on the ICSI corpus (Morgan et al., 2001). The evaluation was revised throughout the project. Chapter 3 includes a comparison of these revisions and an analysis of the overall poor performance of the model.

Due to the poor performance described in Chapter 3, I retrained the model on the ICSI corpus. Chapter 4 outlines how I first failed to use the existing training code from Gillick et al. (2021) and then adapted it using `Lhotse` (zel, 2021), a new audio processing library, which I ended up contributing to Section A.1.

Chapter 5 presents experiments investigating the impact of training data on model performance. It also compares the newly trained models with the original model evaluated in Chapter 3. We find that models retrained on the ICSI corpus perform significantly better than the pre-trained model by Gillick et al. (2021), trained on the Switchboard corpus. However, the performance of the model remains unusable for integration into a video call system. Therefore, Section 5.3 includes some manual investigation of the corpus data. In hindsight, these experiments should have been conducted earlier.

Lastly, Chapter 6 investigates the practicality of the `filtered` option using laughter detection within a video call system; this includes real-time and privacy considerations, as well as alternative solutions. The chapter concludes with suggestions for future work and a latency estimate of the model used in this thesis for future reference.

Throughout the report I regularly mention discussions. These refer to the biweekly meetings with my first and second supervisor, where I presented my progress. For each meeting I prepared a set of slides which are publicly available as part of my public honours project repository (Wolter, 2022b). Wolter (2022a) contains the published laughter recognition codebase.

Chapter 2

Applause and Laughter Detection

Automatic applause and laughter detection is not a new idea. However, research in these two areas is usually done separately; there are a few papers that cover both. One of the most popular papers is Cai et al. (2003) who used sound effect detection - which included laughter and applause - for video summarisation and highlight extraction. Apart from this and a few other papers, most research only covers one of the two domains, either applause or laughter. Thus, the review covers them in separate sections. A third section on evaluation metrics was added to introduce some of the core metrics used in this field.

2.1 Laughter Detection

The research on the acoustic features of laughter reaches back three decades (Bickley and Hunnicutt, 1992). In the early 2000s, the first attempts at laughter detection in conversational speech classified presegmented audio data (Kennedy and Ellis, 2004; Truong and Leeuwen, 2005). These models could only state whether laughter occurred within a given segment. The determination of the segment boundaries of laughter events was not considered. Corpora used in these papers include the development data from the 2004 Spring NIST Rich Transcription Evaluation (Garofolo et al., 2004), the Dutch CGN corpus (Oostdijk et al., 2000) as well as the ICSI Meeting Recorder corpus (Morgan et al., 2001).

Two examples of such classification of presegmented audio data are Kennedy and Ellis (2004) and Truong and Leeuwen (2005). Although both of these models classified presegmented audio, there were some significant differences. Firstly, the definition of laughter and the motivation behind the research differed. While Kennedy and Ellis (2004) define laughter events as ‘points in the meeting where more than one person laughs’, Truong and Leeuwen (2005) considered one person laughing as a laughter event. Kennedy and Ellis (2004) did not specify a particular motivation, whereas Truong and Leeuwen (2005) stated the long-term goal of investigating ‘paralinguistic events’ - laughter being one of them - to classify the speaker’s emotional state.

Secondly, there was a difference between the best performing features and predictors. Features tried include Mel Frequency Cepstral Coefficients (MFCCs), delta MFCCs (MFCC derivatives), spatial cues, modulation spectrum features and Perceptual Linear Prediction(PLP) features. Kennedy and Ellis (2004) got the best results using MFCCs as features and a Support Vector Machine (SVM) for decision making. In contrast, Truong and Leeuwen (2005) used Perceptual Linear Prediction (PLP) features with Gaussian Mixture Models (GMM) for classification.

Truong and Van Leeuwen (2007) continued their research and stated that spectral features alone (such as PLP and MFCCs) can be used to discriminate between laughter and speech, but a significant improvement can be achieved when using prosodic features - features relating to rhythm and intonation.

In 2006, Knox (2006) was the first to perform laughter recognition without the need for presegmented audio data. Their goal was to obtain accurate interval boundaries of laughter segments. One limiting factor for the precision of these boundaries is the frame size. When Knox (2006) first experimented with SVMs similar to Kennedy and Ellis (2004), they realised that the time to compute the features and train the SVM increased significantly with decreasing frame size. This is because aggregate statistics need to be calculated and stored for each frame of training data. On the contrary, a neural network trained with features from a context window can directly use the features of each frame without aggregating them.

To obtain frame-level training labels for the ICSI dataset, Knox (2006) first removed all segments that contained both speech and laughter within one transcription segment, because the segment boundaries of such laughter occurrences could not be identified. The remaining audio data could be accurately separated into laughter and non-laughter segments based on the ICSI transcriptions. Each of these segments was divided into non-overlapping 10ms frames and labeled accordingly.

The model was trained to classify each 10ms frame using a window of 75 frames around the target frame, 37 before and 37 after the frame (Figure 2.1). This way an accurate prediction of laughter boundaries of up to 10ms was possible. The features investigated by Knox and Mirghafori are MFCCs, AC PEAK, F0 and RMS as well as their corresponding delta and delta-delta features (first and second derivatives of the features). They first trained four separate neural networks, each one taking one of the mentioned features as input. Afterwards, they combined the different models by using their output as input for another, smaller neural network. The best results were obtained by combining the outputs of three NN-systems, which used delta MFCCs, AC PEAK and F0 as features, respectively. By training and evaluating separate neural networks for each type of feature first, Knox and Mirghafori observed that MFCCs have the most discriminative power which aligns with prior research by Kennedy and Ellis (2004).

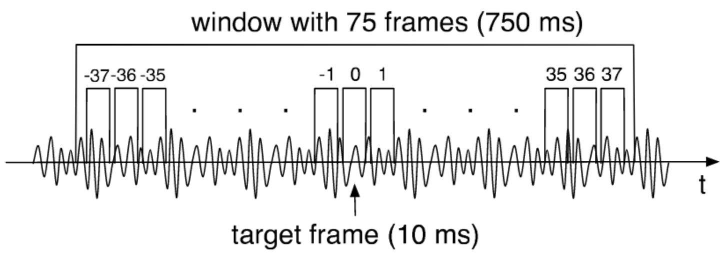


Figure 2.1: Frame window used by Knox (2006)

A survey by Cosentino et al. (2016) investigated research on laughter and laughter detection in different fields up to that time. This survey covers different detection methods using visual, acoustic and sensory data. Cosentino et al. (2016) found that when using solely acoustic features best performance is obtained by using MFCCs and PLPs as features and combining the resulting classifier with ones that also use prosodic features. This aligns with findings from prior research (Truong and Van Leeuwen, 2007; Knox, 2006).

More recent research by Gillick et al. (2021) finds that features learnt from spectrograms using deep convolutional networks outperform previous approaches based on MFCCs. They hypothesised that models using MFCCs are more prone to pick up surface level characteristics of sound and thus, will be more sensitive to variations like background noise. Gillick et al. (2021) expected that features learnt directly from the spectrogram are more representative of actual laughter and, thus, more robust to different environments. To test this hypothesis, they trained three different models: a baseline feed-forward neural network with MFCC features from their existing research (Ryokai et al., 2018), a ResNet-18 (He et al., 2016) model working directly on spectrogram data and the same ResNet model using augmented features. Augmentations applied include pitch-shifting, time-stretching, reverberation and SpecAugment (Park et al., 2019). SpecAugment is a simple method to augment spectrograms by time warping, frequency masking, and time masking. Figure 2.2 shows that frequency and time masking is done by blocking out certain spectrogram regions, vertically or horizontally.

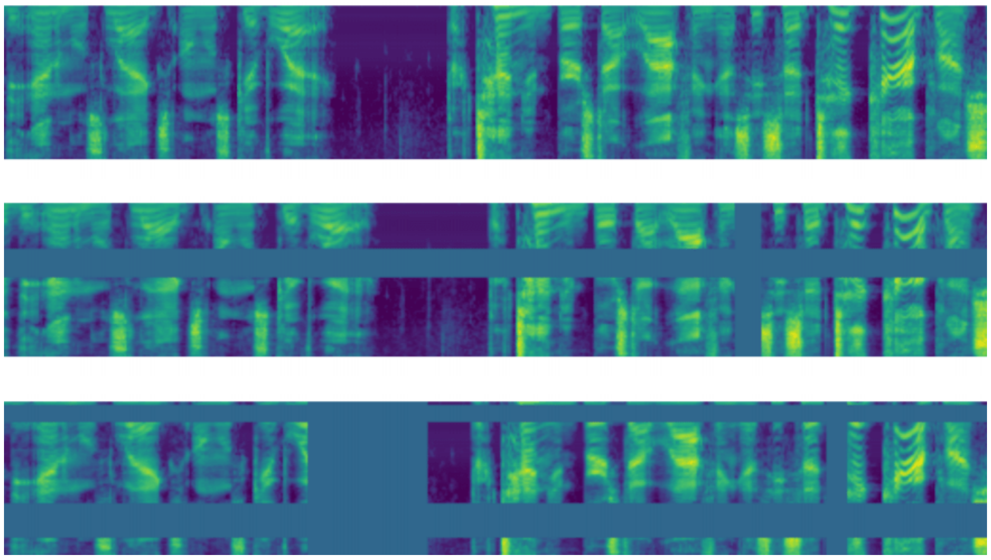
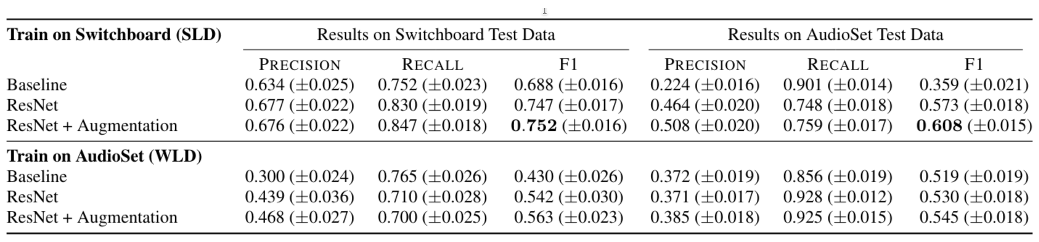


Figure 2.2: SpecAugment example

Gillick et al. (2021) trained their model on two corpora: the Switchboard corpus (Godfrey et al., 1992) and AudioSet (Gemmeke et al., 2017). They used Binary Cross Entropy loss for optimisation, which is explained in Section A.2. In contrast to prior work, they also evaluated the models on the corpus that it was not trained on. This mismatch between training and testing data tries to resemble the difference between clean audio and real-world test data.

Their results (Figure 2.3) mention strongly and weakly labeled data. Weakly labeled means that only the presence or absence of an event is given, whereas strongly labeled data states precise boundaries. AudioSet (Gemmeke et al., 2017) is a weakly labeled corpus consisting of ten second segments sampled from YouTube videos. Each segment has one or more annotations like `laughter` or `wind`, which state that these sounds are present in the given segment. Since segments are sampled from a range of YouTube videos, domains in this corpus vary greatly. The Switchboard dataset is a strongly labeled corpus consisting of telephone conversations between two individuals. Compared to AudioSet, the domain is homogeneous across all recordings and a transcription file provides precise boundaries for audio events.

Figure 2.3 shows that in all evaluations, the baseline model using MFCC features and a feed-forward neural network is outperformed by the ResNet architecture in every metric, except the recall for the model trained on AudioSet and evaluated on the Switchboard test data. Gillick et al. (2021) conclude that their ResNet-based model provides a robust, state-of-the-art laughter-detection algorithm that mitigates some of the problems of noisy real-world environments. This is one of the reasons why it is used as a basis for this project.



Train on Switchboard (SLD)	Results on Switchboard Test Data			Results on AudioSet Test Data		
	Precision	Recall	F1	Precision	Recall	F1
Baseline	0.634 (±0.025)	0.752 (±0.023)	0.688 (±0.016)	0.224 (±0.016)	0.901 (±0.014)	0.359 (±0.021)
ResNet	0.677 (±0.022)	0.830 (±0.019)	0.747 (±0.017)	0.464 (±0.020)	0.748 (±0.018)	0.573 (±0.018)
ResNet + Augmentation	0.676 (±0.022)	0.847 (±0.018)	**0.752** (±0.016)	0.508 (±0.020)	0.759 (±0.017)	**0.608** (±0.015)
Train on AudioSet (WLD)						
Baseline	0.300 (±0.024)	0.765 (±0.026)	0.430 (±0.026)	0.372 (±0.019)	0.856 (±0.019)	0.519 (±0.019)
ResNet	0.439 (±0.036)	0.710 (±0.028)	0.542 (±0.030)	0.371 (±0.017)	0.928 (±0.012)	0.530 (±0.018)
ResNet + Augmentation	0.468 (±0.027)	0.700 (±0.025)	0.563 (±0.023)	0.385 (±0.018)	0.925 (±0.015)	0.545 (±0.018)

Figure 2.3: Results by Gillick et al. (2021) with 95% confidence intervals. Precision, Recall, and F1 scores are calculated as segment-based metrics (Section 2.3.1). SLD=Strongly labeled data, WLD=weakly labeled data.

2.2 Applause

The following review is less exhaustive because this thesis focuses on laughter detection. Applause detection is a possible extension of the project to meet the original goal mentioned in Chapter 1. More information on this decision can be found (Section 3.1).

As mentioned at the beginning of this chapter, in 2003 Cai et al. (2003) worked on the detection of sound events, including applause. They used perceptual features and MFFCs as features, Hidden Markov Models (HMM) and Gaussian Mixture models (GMM) to model sound and log likelihood for decision making. Cai et al. (2003) evaluated their system on a testing set consisting of 2 hours of video material from different programs. For applause specifically, they achieved a precision of 87.37% and a recall of 92.00%.

In contrast, Uhle (2011) used MFCCs and low-level descriptors (LLD) to represent sound and passed these features to an Multi-Layer-Perceptron (MLP) or an SVM for classification. The difference between the MLP and SVM classifier was rather small. Uhle worked with a relatively small dataset consisting of 210 segments of 9-30s length. This equates to a total length between 31.5 min and 105 min. On a rather small test (10% of this dataset) Uhle achieved an accuracy of 95% with a precision of 96.51% and recall of 97.65%.

A less complex approach for applause detection was presented by Li et al. (2009). They used a manually created 4-layer decision tree to classify a given sound input as applause or not-applause. Testing their model on 50 hours of meeting speech containing 500 applause segments ranging from 0.8s to 36s they were able to retrieve 491 of the 500 applause segments while incorrectly retrieving 38 non-applause segments. This equates to a recall of 98.2% and precision of 92.82%. Li et al. also compared their less complex model to the HMM model proposed by Cai et al. (2003) and outperformed it while using less computational time.

Manoj et al. (2011) proposed another approach based on manually created decision trees. They also compared it to a more complex model similar to the one proposed by Cai et al. (2003) - with the difference that this model only used GMMs, no HMMs. Even though the decision tree stages were different to Li et al. (2009) the findings are similar. The decision tree outperforms the more complex method using MFCCs and GMMs.

2.3 Evaluation Metrics

2.3.1 Accuracy, Precision and Recall

Accuracy, precision and recall are standard metrics for performance evaluation. All three metrics are calculated by comparing the predicted class labels to the true class labels. This section states the formulae, their meaning for our use case and the advantages of one metric over the other. Let:

1. TP: True Positives - laughter samples correctly classified as laughter,
2. FP: False Positives - non-laughter samples incorrectly classified as laughter,
3. TN: True Negatives - non-laughter samples correctly classified as non-laughter,
4. FN: False Negatives - laughter samples incorrectly classified as non-laughter.

Accuracy: Percentage of correctly classified samples (laughter and non-laughter) over the total number of samples.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Precision: Percentage of correctly classified laughter samples over all samples predicted as laughter.

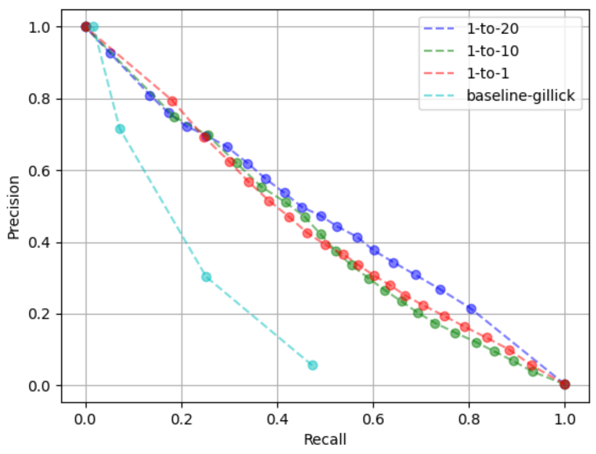
$$Precision = \frac{TP}{TP+FP}$$

Recall: Percentage of correctly classified laughter samples over all laughter-samples.

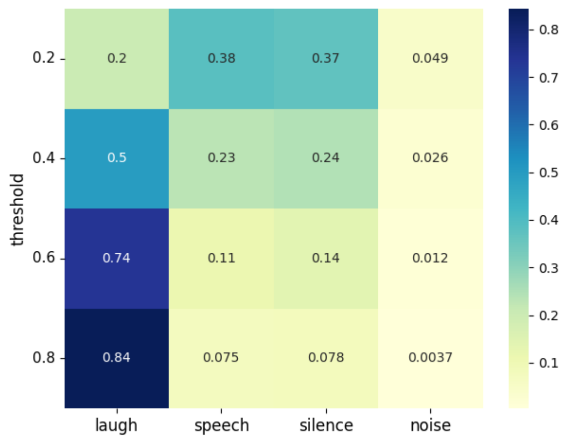
$$Recall = \frac{TP}{TP+FN}$$

For our use case, we are interested in continuous audio data, but all metrics presented above are calculated on discrete data points. We make use of segment-based metrics suggested by Mesaros et al. (2016) which split a continuous audio stream into fixed-sized segments (e.g. 10ms). These fixed-sized segments are treated as separate instances and build the base-unit for our evaluation. With 10ms segments, 1s laughter will become 100 units of laughter. Assuming that 0.6s are predicted correctly this would yield 60 TP and 40 FP. Gillick et al. (2021) used the same approach for their evaluations.

Accuracy is a good metric if classes are balanced. In our use case, the two classes are highly imbalanced; laughter only occurs a few times in a long audio stream. If we choose a very high threshold the model predicts all segments as non-laughter. This leads to a large number of true negatives which skews the accuracy. This model with no skill achieves high accuracy. Considering the same example, the precision will be 100% as well, but the recall will be 0% because the model did not retrieve any laughter segments correctly. The other extreme is given by a threshold of 1. The model will predict everything as laughter, leading to low accuracy and low precision because only the few seconds of laughter in the whole meeting are classified correctly. The recall will be 100% because the model retrieved all laughter that occurs. To capture the range between these two extremes, a precision-recall curve is a useful tool. Precision recall curves plot the precision against the recall for a range of thresholds (e.g. Figure 2.4a). This provides a more detailed view of the model performance and answers questions like: ”What is the precision of the model if we require a minimum recall of 80%”. To conclude, for our use case, reporting precision and recall across a range of thresholds is a more expressive metric than accuracy.



(a) Example precision recall curve (Section 2.3.1)



threshold	laugh	speech	silence	noise
0.2	0.2	0.38	0.37	0.049
0.4	0.5	0.23	0.24	0.026
0.6	0.74	0.11	0.14	0.012
0.8	0.84	0.075	0.078	0.0037

(b) Example confusion matrix (Section 2.3.2)

Figure 2.4: Two types of visualisations used in this report.

2.3.2 Confusion Matrix

A confusion matrix is a good way to identify misclassifications. In a multiclass problem, it shows the true label on one axis and the predicted label on the other. It can either show the actual counts of predictions, in terms of TP, FP, TN and FN as described in Section 2.3.1, or display them as normalised values between 0 and 1.

For laughter recognition, we only have two classes. Instead of plotting a 2x2 confusion matrix, evaluations in this thesis use a different approach. Since non-laughter segments can be split into subclasses (as we will see in Table 3.1), we plot the percentage of misclassified laughter samples for each of these classes. Each row in Figure 2.4b represents the model outputs for a certain threshold and shows the percentage of true laughter samples classified as laughter, speech, silence and noise, respectively. Thus, the sum of all values in a single row is one and the the column of laughter is the precision of the model at this threshold. Note that we cannot get the same diagram for non-laughter segments because the model is a binary classifier. It only predicts laughter and non-laughter but none of the subclasses.

Chapter 3

Evaluation of Existing Models for Laughter Detection

3.1 Model and Data Selection

After the initial research, we decided to focus solely on laughter detection. There are two main reasons for this: the type of previous research and open source code available. As mentioned in Chapter 2 most research in the field of laughter and applause detection is done separately. Thus, implementing a detection algorithm from existing research requires merging different models. For a combined model to work, a good understanding of both domains is essential. We decided that this is most easily attained by focusing on one domain first. We chose to focus on laughter instead of applause because there was an open source repository available that showcased an existing state-of-the-art laughter detection algorithm (Jrgillick, 2021; Gillick et al., 2021).

The domain in video conferences is separate single person audio tracks usually recorded in a relatively quiet environment. To investigate how the existing model might perform in this domain, we decided to evaluate it on the ICSI meeting corpus (Morgan et al., 2001). We initially considered the use of AudioSet (Gemmeke et al., 2017) as a possible evaluation corpus. We discarded this idea because of a mismatch with the domain of our project. AudioSet consists of 10s audio segments from Youtube videos which were labeled according to a fixed set of classes - including laughter. Nevertheless, the domain of Youtube videos varies drastically. It contains recordings of someone laughing in a silent environment inside a room and someone laughing in a very noisy environment outside. This variety of environments might help for generalising laughter detection but is not suitable for our project, as we focus on laughter in online meetings. In contrast, the ICSI corpus is a good match to our domain as it solely consists of meeting speech recorded with both close-distance and table-top microphones. We only make use of the close-distance microphone recordings which capture the audio of a single meeting participant. The ICSI corpus provides transcriptions for all its participants containing speech as well as `Vocal` and `NonVocal` tags for sounds like laughter and door squeaks, respectively. There are two further reasons for using the ICSI corpus: its size and its licence. The ICSI corpus has a total length of 72h and around 3.9h of total laughter duration suitably transcribed for our use case (Section 3.2). The licence is CC BY 4.0 which means that it can be used for commercial projects like an improved version of video meeting software.

3.2 Preprocessing

Before the model by Gillick et al. (2021) could be evaluated on the ICSI corpus, some preprocessing of the corpus data was needed. At first, the 72 meetings were split into training, development and test set to minimise speaker overlap. The partitioning follows the structure from Renals and Swietojanski (2014). It uses meetings `(Bmr021, Bns001)` as development set, meetings `(Bmr013, Bmr018, Bmr021)` as test set and the remaining 67 meetings as training set. After this, all laughter segments needed to be filtered out from the raw data to accurately determine correct classifications. Lastly, I used the ICSI-transcriptions to divide non-laughter segments into subclasses to get a more rigorous analysis of misclassifications.

3.2.1 Filtering out Laughter-only Segments

ICSI transcriptions are split into segments denoted by a start and end time. Consequently laughter segments co-occurring to speech in one segment cannot be precisely identified. As mentioned in Section 2.1 such data is called weakly labeled. Strongly labeled data is key for this project because our model should identify laughter segments in a continuous audio stream. Filtering out strongly labeled laughter segments resembles the approach by Knox (2006) to obtain frame-level training data (Section 2.1). All weakly labeled segments containing laughter are categorised as invalid and disregarded from the evaluation, a total of 3765 segments. The remaining 8415 laughter segments have a total duration of 3.9h. Table 3.1 shows how this compares to the fraction of other types of audio in the ICSI corpus.

Alternatively, I could have created an ASR model to align weakly-labeled laughter events with their audio counterpart. I did not follow this approach due to lack of experience with ASR, which meant that creating such a model would have been time-consuming. I considered the 3.9h of pure laughter as sufficient for this project and decided to focus on other parts of the project. In general, more training data is preferable. Thus, future projects might consider this approach to gain strongly labeled data for the whole corpus and make use of the additional laughter occurrences contained in the invalid segments.

subclass	number of segments	total duration
laughter	8415	3.9 h
invalid	3765	3.7 h
speech	98418	64.1 h
noise	18224	10.9 h
silence	97431	681.9 h

Table 3.1: Number of segments and total duration of each subclass. Segments can have arbitrary length and are not limited as in Chapter 4.

3.2.2 Subclasses of Non-laughter

In addition to the obvious class division between laughter and non-laughter, we can divide segments into smaller subclasses. This becomes useful when evaluating classification (as we will see in Chapter 5). For non-laughter segments we can distinguish between speech, invalid segments (as in Section 3.2.1), noise and silence. To summarise, there are five subclasses:

1. **laughter**: segments only containing laughter
2. **non-laughter**
   (a) *speech*: segments only containing speech
   (b) *invalid*: segments where laughter occurs together with other sounds (see Section 3.2)

(c) *noise*: segments containing vocal-sounds like coughing, non-vocal sounds like chair squeaks or a mixture of sounds and speech in one segment

(d) *silence*: segments for which no transcription exists

Note that these subclasses of non-laughter are not classified by the model. The model is a binary classifier. The distinction within the non-laughter class is merely used for evaluation. Thus, the confusion-matrices in Section 5.1.2 are only for laughter misclassifications as described in Section 2.3.2. The breakdown of the amount of each subclass is shown in Table 3.1.

3.3 Evaluating Gillick et al's Model on the ICSI Corpus

In contrast to some existing research (Kennedy and Ellis, 2004; Knox, 2006) which only used a subset of the ICSI corpus, I evaluated the whole ICSI dataset. I ran the pre-trained detection model by Gillick et al. (2021) with four different thresholds for each channel of the 75 meetings - a total of 468 audio tracks. Over the course of this project, I corrected the evaluation two times. The following sections compare the results of these evaluations and provide some analysis on the poor performance of the model.

3.3.1 Comparing the Initial and Corrected Evaluations

In the initial evaluation (Table 3.2), the maximum recall achieved was 31.53%. I expected that a low threshold like 0.2 would yield low precision. Getting such a low recall, however, was surprising.

threshold	precision	recall
0.2	14.88%	31.53%
0.4	33.26%	17.85%
0.6	49.81%	8.05%
0.8	63.08%	3.01%

Table 3.2: Initial evaluation on 468 audio tracks for different thresholds.

After some discussion, I realised that, even though I filtered out invalid laughter segments from the raw ICSI data as described in Section 3.2.1, I did not exclude them from the predictions before evaluation. This meant that the model could predict a laughter occurrence correctly but the evaluation method would consider it as false because this laughter occurrence was discarded during preprocessing. This discrepancy in preprocessing was easily fixed and improved the precision significantly (Table 3.3). However, the recall did not increase by much.

	initial eval		considering invalid segs	
threshold	precision	recall	precision	recall
0.2	14.88%	31.53%	20.02%	37.21%
0.4	33.26%	17.85%	48.39%	20.74%
0.6	49.81%	8.05%	73.68%	9.14%
0.8	63.08%	3.01%	91.44%	3.33%

Table 3.3: Corrected evaluation considering invalid segments compared with the initial evaluation (Table 3.2).

Lastly, the final evaluation (Table 3.4) includes two significant changes: altering the model to remove a filter and introducing a weighted average, as we now explain. The evaluation code used by Gillick et al. (2021) sometimes yielded probabilities smaller than 0 and larger than 1. This bug was caused by a low-pass filter implemented in `laugh_segmenter.py`. I did not investigate this further because neither the code nor the paper stated why this filter was necessary. I fixed the issue by removing the filter.

	initial eval		considering invalid segs		final eval	
threshold	precision	recall	precision	recall	precision	recall
0.2	14.88%	31.53%	20.02%	37.21%	20.86%	45.58%
0.4	33.26%	17.85%	48.39%	20.74%	50.26%	27.77%
0.6	49.81%	8.05%	73.68%	9.14%	73.77%	13.35%
0.8	63.08%	3.01%	91.44%	3.33%	84.53%	4.76%

Table 3.4: Final evaluation on 468 audio tracks, compared to the initial (Table 3.2) and corrected evaluation (Table 3.3).

Further, introducing a weighted average had a significant impact on the evaluation. The first two evaluations in Table 3.2 are calculated by averaging the precision and recall of each meeting. These results can be misleading because meetings vary in duration and some contain significantly more laughter occurrences than others. Figure 3.1 shows

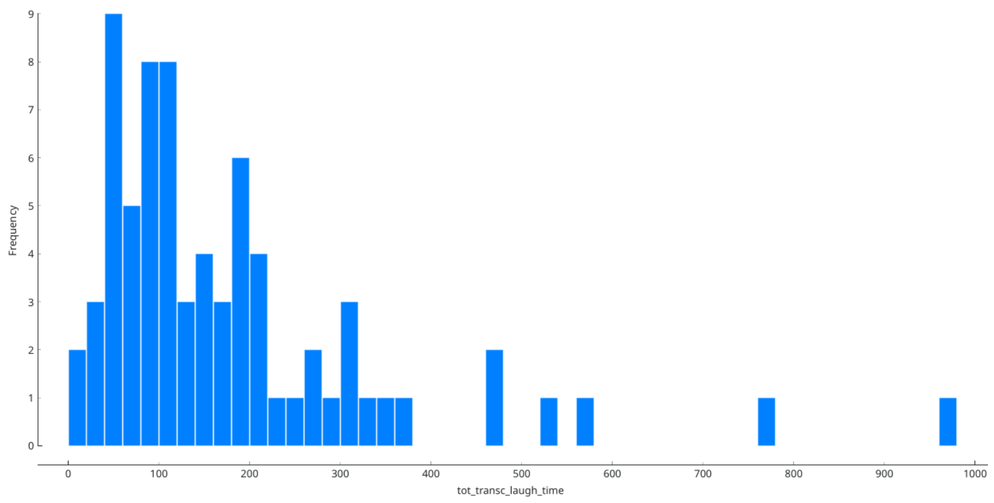


Figure 3.1: Distribution of transcribed laughter duration per meeting [in s]; bin-width=20s.

that the total duration of laughter occurrences ranges from $< 20\mathrm{s}$ to $> 950\mathrm{s} = 15\mathtt{min}$. For our use case, it makes sense to take the weighted average because we are interested in the total duration of correctly and incorrectly predicted laughter events, the artificial unit of meetings is an unimportant distinction to us.

To conclude, the final evaluation shown in Table 3.4 should be considered correct and is used as a baseline for further experiments (as we will see in Chapter 5).

3.3.2 Analysing Poor Model Performance

To get a better understanding of the performance, Table 3.5 shows what the evaluation would correspond to if this model was implemented as part of a video call system. We assume a meeting that lasts 1 hour and contains a total duration of 4 minutes of laughter. Using any of the investigated thresholds would yield an unusable system. We will briefly examine the two extreme cases. If the threshold was really low, e.g. 0.2, 1 minutes and 49 seconds of laughter would be correctly retrieved. Despite this being not even half of the laughter events that occurred we also retrieve almost 7 minutes of noise. On the other hand, choosing a high threshold, e.g., 0.8, yields very high precision and thus, only two seconds of noise are transmitted. However, the system merely retrieves 14 seconds of the four minutes of laughter. With such a low percentage of retrieved laughter events, there is no significant difference to participants muting themselves entirely.

threshold	precision	recall	predicted	actual laughter	noise
0.2	20.86%	45.58%	8:44 min	1:49 min	6:56 min
0.4	50.26%	27.77%	7:16 min	2:01 min	5:15 min
0.6	73.77%	13.35%	0:43 min	0:32 min	0:11 min
0.8	84.53%	4.76%	0:14 min	0:12 min	0:02 min

Table 3.5: Practical example using values from the final evaluation column in Table 3.4.

A possible reason for such a bad performance on the ICSI corpus is a data mismatch between training and evaluation data. The pre-trained model by Gillick et al. (2021) was trained on the Switchboard corpus (Godfrey et al., 1992). The Switchboard corpus records telephone conversations between two participants at a sample rate of 8000 hz. The ICSI corpus records meeting speech of multiple participants at a sample rate of 16000 hz. For prediction, all audio tracks are down-sampled to 8000 hz, which results in losing lots of the original data. Furthermore, the Switchboard dataset has one audio track containing the audio from both participants, whereas the ICSI corpus has separate audio tracks for each participant. Thus, especially the long periods of silence on each individual channel are new to the model.

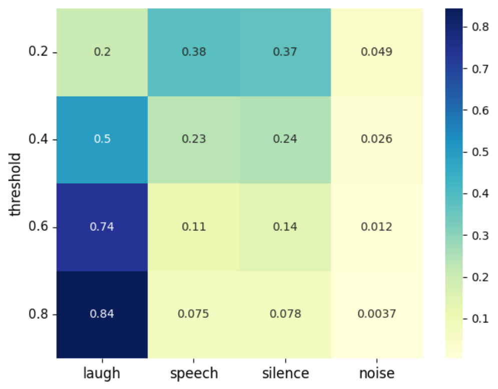


Figure 3.2: Confusion matrix for final evaluation (Table 3.4), evaluated on the whole ICSI corpus. Each row shows the percentage of laughter segments classified as a certain subclass for a given threshold (see Section 2.3.2).

Looking at the confusion matrix in (Figure 3.2) we notice that the misclassifications are almost evenly split between silence and speech regions. This is surprising because I expected silence to be easier to distinguish from laughter than speech. We will see further investigations of these silence regions in autorefcha:experiments. In response to this finding, I decided to retrain the model on the ICSI corpus.

Chapter 4

Training Laughter Detection on the ICSI Corpus

4.1 Adapting Gillick et al.'s Training Code

The first approach was to adapt the code published by Jrgillick (2021), which used PyTorch (Paszke et al., 2017), an open source machine learning framework for Python. Parts of the code were cumbersome and disregarded the standards of PyTorch programs, e.g., they used a manual batch counter for training steps instead of using epochs.

Despite these concerns, I decided to adapt their existing code for two reasons. Firstly, developing the whole framework from scratch would have been time consuming and slightly precarious as I had never worked on a machine learning project of this scale. Secondly, slightly adjusting the code allows for an comparison between the original and retrained model.

Figure 4.1 shows the data pipeline used by Gillick et al. (2021). The raw inputs per conversation are one audio track and two transcription files. Firstly, all audio tracks are preloaded into memory, serialised and stored as a hash map. The keys of this hash map are the audio paths present in the dataframe shown in Figure 4.1, the values are the serialised audio tracks. Every time the model is trained, the hash map is loaded into memory to quickly access the audio tracks. The size of this hash map is investigated later in this section. Secondly, Gillick et al. (2021) filter out all laughter segments and store them in a dataframe as shown in Figure 4.1. They add the same amount of non-laughter segments to this dataframe. Note that dataframes are basically tables.

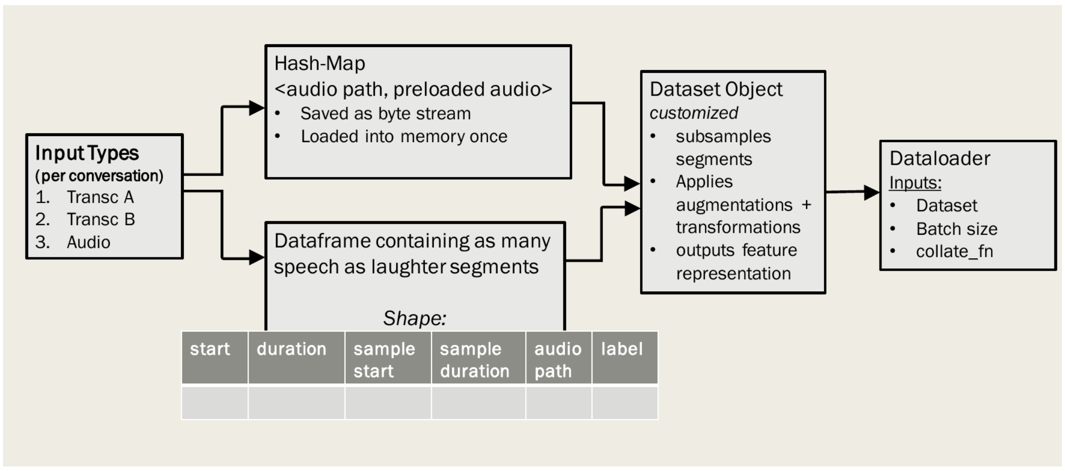


Figure 4.1: Data pipeline used by Gillick et al. (2021). I created this pipeline based on Gillick et al's code (Jrgillick, 2021). This diagram was not part of the original paper.

The term originates from the commonly used data analysis Python library pandas (Reback et al., 2021). Each segment is stored with its start time, duration, subsample start time, subsample duration, audio path and label, where 1 means laughter and 0 means non-laughter. This dataframe and the hash map are the inputs for a PyTorch Dataset. Dataset here does not refer to actual data but to one of the two main PyTorch primitives for data handling. A PyTorch Dataset defines the data structure for the PyTorch DataLoader, which is used to sample batches during training.

Creating a hash map with serialised audio data from the ICSI corpus failed due to missing memory. Since the ICSI corpus contains single audio tracks for each participant the total size is larger than the Switchboard dataset. In addition, the sample rate is twice as high.

The following estimates show the difference in the storage space needed to preload the whole ICSI/Switchboard corpus as .wav-files. We assume a bit rate of $16\ \mathtt{bits} = 2\ \mathtt{bytes}$ per sample.

The Switchboard corpus consists of 260 hours of raw audio sampled at $8\ \mathtt{khz}$, which can be estimated as:

$$\frac{260 \text{ hours} \cdot 3600 \text{ seconds} \cdot 8000 \text{ samples per second} \cdot 2\ \mathtt{bytes}}{1024^3} \approx 13.95\ GB$$

The ICSI corpus contains 72 hours of meeting speech with an average of six participants per meeting. Thus, there are $72 \cdot 6 = 432$ hours of audio sampled at $16\ \mathtt{khz}$:

$$\frac{432 \text{ hours} \cdot 3600 \text{ seconds} \cdot 16000 \text{ samples per seconds} \cdot 2\ \texttt{bytes}}{1024^3} \approx 46.35\ GB$$

Using a machine with sufficient RAM would allow for loading all of the audio data into memory. Regardless of the lack of access to such a machine, finding a solution that requires less RAM was desirable. Loading all raw audio into memory, even though the model only needs the feature representation of short segments, seemed like a waste of resources. Therefore, I decided to adapt the data pipeline, which turned out to be more difficult than expected and took me 1.5 months. The following two sections outline the two different approaches that I tried.

4.2 Rewriting the DataLoader from Scratch

The first approach was to amend the data-pipeline in Figure 4.1 by removing the hash map. Since the hash map is one input to the Dataset, which is used by the DataLoader, those two structures had to be rewritten as well. The adapted pipeline was slow and only allowed the model to train around 30 batches of audio in two hours, each consisting of 32 audio tracks. With training samples of one second each, this equates to one sample every eight seconds. This training performance, measured on a GPU machine, was significantly slower than evaluating the pre-trained model from Chapter 3 on a CPU-only machine. GPUs are optimised for matrix computations, which are the basis of most machine learning algorithms. Thus, using a GPU should normally be significantly faster than using the CPU.

It turned out that the GPU was not used most of the time because the data loading was the bottleneck. In contrast to inference, where a whole audio track is loaded once at the beginning, training required the DataLoader to load many one-second segments from different offsets. Without any preloading in place, the segments were loaded on-the-fly, requiring one disk access per one-second segment. This meant the GPU was starved, i.e. waited for input most of the time.

Additionally, loading data from large offsets using the librosa library (McFee et al., 2015), a popular Python library for audio analysis, was significantly slower than loading data from the beginning of an audio file. A crude analysis (Table 4.1) shows that the loading time grows with the offset at which a sample is located. This is especially problematic because ICIS's average meeting length of 58 minutes (LDC, n.d.a) is significantly longer than the 6.5 minutes for Switchboard recordings (LDC, n.d.b). Later, I realised that this problem only arose because ICSI audio files are stored as `.sph`-files. Converting all files to `.wav`-files could have solved this issue, but the main bottleneck would have persisted: the large number of disk accesses due to on-the-fly data loading.

offset[s]	average time[s]	total time[s]	iterations run
0	0.15	4.40	30
300	0.34	10.16	30
700	0.59	17.72	30
1000	0.78	23.36	30
3000	2.03	60.79	30
5000	3.26	97.65	30

Table 4.1: Using librosa (McFee et al., 2015) to load a one-second segment from different offsets of an audio-file. Channel loaded: `Btr002:chan3`. Duration: 5318s. Test code: naive librosa-test.

My second supervisor suggested the use of Lhotse (zel, 2021) to speed up data loading. Lhotse is a Python library for speech and audio data preparation that is still in development. The following section outlines this seconds approach of adapting the data pipeline.

4.3 A Data Pipeline Using Lhotse

This section describes how I used Lhotse to create a data pipeline with faster data loading. Instead of trying to stick to the design of Gillick et al. (2021) (Figure 4.2), I decided to combine the features provided by Lhotse with some of the code that I had already written. In this section I first give a brief introduction into Lhotse and why it is suitable for our use case. Then I explain some necessary adjustments and finally present my data pipeline which is part of the published codebase.

4.3.1 Lhotse

Three main features make Lhotse suitable for my use case:

1. Simplified loading of popular corpora like ICSI

2. Sample creation based on metadata without loading raw audio data from disk (explained below)

3. Efficient I/O (we will see that in Section 4.3.2)

Lhotse provides so-called *recipes* which simplify the work with popular corpora such as the ICSI corpus. These *recipes* are Python scripts that create *manifests*. *Manifests* are representations of the entire corpus that contain metadata of the recordings and the corresponding transcriptions, which Lhotse calls *supervisions*. Using those *manifests* Lhotse can define audio samples - called *Cuts* - solely based on metadata without loading raw audio or transcript data from disk. Actual data loading is done lazily, only when it is needed, e.g. during feature computation. A list of such *Cuts* is called *CutSet* and is the third important data structure used in the data pipeline, in addition to pandas dataframes and Lhotse *manifests*.

Due to Lhotse being in development, the *recipe* for the ICSI corpus was not part of the latest release *v0.12*. Hence, I had to work with the development version and had to fix some bugs in the ICSI *recipe*. I ran into some other issues with Lhotse and decided to debug some of them. Even though some of the debugging was not strictly necessary for my project, I decided to contribute to this open source project which has the potential to be useful in the field of audio and speech processing in the future. A list of my contributions, containing links to the corresponding pull requests, can be found in Section A.1. All contributions have been accepted and are now part of the Lhotse library.

4.3.2 Data Pipeline

Figure 4.2 shows the data pipeline. Initially, Lhotse's *ICSI-recipe* takes the raw data (in red) and creates a *manifest* representing the ICSI corpus, as described in Section 4.3.1. Besides this manifest creation there are two main functions in the data pipeline: `create_data_df` and `compute_features`.

`compute_features` has two parts. The first part uses the *ICSI-manifest* to create a mel spectrogram for the whole duration of each audio track in the corpus. These feature representations are stored as *CutSets*, one for each split. Manifest creation and feature computation for the whole corpus, marked in blue, only need to happen once. Note that if the feature representation is changed,e.g., the dimensions of the spectrogram, this part of `compute_features` needs to be run again to update the features of the whole corpus.

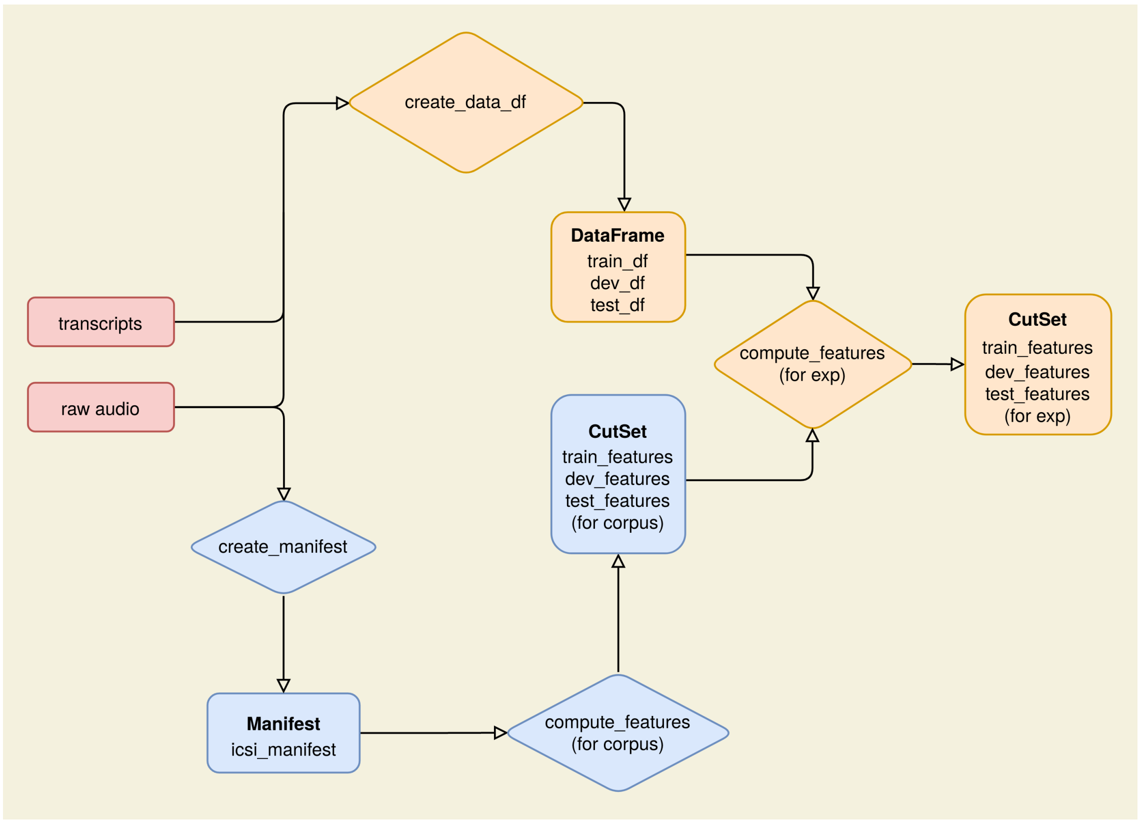


Figure 4.2: Data Pipeline. Rectangles represent data (with structure type in bold), diamonds represent functions. The arrows show input and output of these functions. `df` stands for dataframe.

The remaining parts, marked in orange, need to be executed every time the model is trained on a new subset of the corpus. `create_data_df` creates three dataframes of the shape shown in Figure 4.1, one for each split. These dataframes contain all information needed to create the final *CutSets*, which consists of the features and the *supervisions* for each sample in the training data. `compute_features` creates these *CutSets* in two steps. For each row in the dataframe it takes the feature representation of the corresponding audio track from the corpus *CutSet* (in blue) and extracts the part defined by subsample start and subsample duration, which are fields of the orange dataframe. Since `compute_feautures` only extracts parts of already existing features and does not recompute anything, this is guaranteed to be fast.

Then, `compute_features` uses the label from this dataframe row to create a custom *supervision segment* stating whether this sample is laughter or not. This *supervision segment*, the feature representation and some other meta data like the channel and meeting id are stored together in a *Cut* and provide the final representation of one training sample. By applying this process across all rows in the dataframe, we end up with three *CutSets*, one for each split.

The following example demonstrates the improvements of this new data pipeline. A *CutSet* with 170000 *Cuts*, which we will see in Chapter 5, has the size of 1 `GB`. The *CutSets* containing the features for the whole corpus (blue in Figure 4.2) has a size of 21 `GB`. The sum of 22 `GB` would already be significantly lower than the 46.35 `GB` estimated in Section 4.1. However, not all of these 22 `GB` need to reside in RAM during training. Training succeeded on a GPU machine with 8 `GB` RAM. It took 25 minutes to train one epoch with a training set of 170000 one-second samples. This averages to around 113 samples per second. Compared to one sample every eight seconds in the first version of the data pipeline (Section 4.2), this is a massive improvement.

I can not explicitly quantify the memory requirement, as I do not fully understand all the internals of Lhotse, nor can I compare the training speed of my new pipeline to the one from Gillick et al. (2021) because they did not report it. Nevertheless, I conclude that I managed to drastically decrease the memory requirements, avoid data loading and feature computation on-the-fly and attain reasonable training speeds on the ICSI corpus.

Future work can further simplify the pipeline by using the *supervision manifest* created by the *ICSI-recipe* instead of creating custom *supervision segments*. This would make the `create_data_df` function obsolete.

4.4 Training

Given the data pipeline outlined in Section 4.3.2 the actual training was straightforward. For each different subset of the corpus, I needed to rerun the `create_data_df` and `compute_feature` functions to create the features.

I adapted the PyTorch Dataset for voice activity detection (VAD) already provided by Lhotse. The *CutSets* were loaded into this modified VAD Dataset, which I called LAD: laugh activity detection. During training time, data is extracted from this Dataset using the `SimpleCutSampler` provided by Lhotse. To allow for direct comparison, I used the same ResNet architecture as Gillick et al. (2021). The performance of this model trained on different subsets of the ICSI corpus is investigated in the next section.

Chapter 5

Investigating the Impact of Training Data on Model Performance

Representative training data is essential for machine learning. Training on the entire ICSI corpus would take a long time and was not feasible for my project. Thus, the following experiments investigate how the selection of a corpus subset for training impacts the model's performance. Due to significant time spent on the tasks already outlined in the thesis, the following experiments do not include exhaustive grid searches. Hence, the results of these experiments should be considered as pointers for future research, not an optimal solution.

The structure of the training data was investigated in two dimensions:

1. class balance between laughter and non-laughter samples
2. diversity of non-laughter samples

All evaluations in this section are done on the development set of the ICSI corpus as defined in Section 3.2.

5.1 Experiment 1 - Class Balance

The first set of experiments evaluates different ratios between laughter and non-laughter segments in the training data. Three different ratios between laughter and non-laughter segments are compared with the pre-trained model by Gillick et al. (2021): 1-to-1, 1-to-10, and 1-to-20. The sample duration is one second, as also used by Gillick et al.

(2021). The feature shape differs from the pre-trained model. Gillick et al. (2021) use mel spectrograms of shape 44x128, whereas the newly trained models use a shape of 100x40. Furthermore, neither subsampling nor feature augmentation was applied during training, which helped Gillick et al. (2021) to achieve minor performance improvements. I initially planned to investigate these three dimensions further: feature shape, subsampling, and feature augmentation. Due to time constraints, I did not conduct such experiments, which leaves this task for future work.

The non-laughter segments are sampled at random from all non-laughter segments, without considering the subclass. Section 5.2 investigates a fixed distribution between these subclasses.

5.1.1 Method

To select non-laughter segments across a variety of meetings, `create_data_df` (Figure 4.2) uses the following selection process. For each laughter segment, a channel within the same meeting is selected at random. A non-laughter segment of the same duration as the laughter segment is sampled from this channel. If the laughter segment is longer than one second, both laughter and non-laughter segments are truncated to a random one-second region of their segment. If the segment is shorter than one second, it is stored unmodified and padded to one second length during feature computation. All these segments are stored in a dataframe as explained in Section 4.3.2. In contrast to random sampling across all meetings, this approach guarantees that the training data holds non-laughter segments from all meetings containing laughter.

5.1.2 Results

Figure 5.1 compares the performance of the newly trained models with the baseline given by the pre-trained model by Gillick et al. (2021) evaluated on the development set. The data points displayed in Figure 5.1 are precision and recall for 21 evenly spaced thresholds from 0 to 1. For the baseline, there are only four data points, because the data is taken from the evaluation presented in Chapter 3. All models trained on the ICSI corpus outperform the baseline. Overall, there is little difference between the three new models, but for `recall` > 0.3 the 1-to-20 model performs best.

Figure 5.2 shows a confusion matrices for different thresholds to deepen the understanding of misclassified laughter samples. Note that the confusion matrix for the

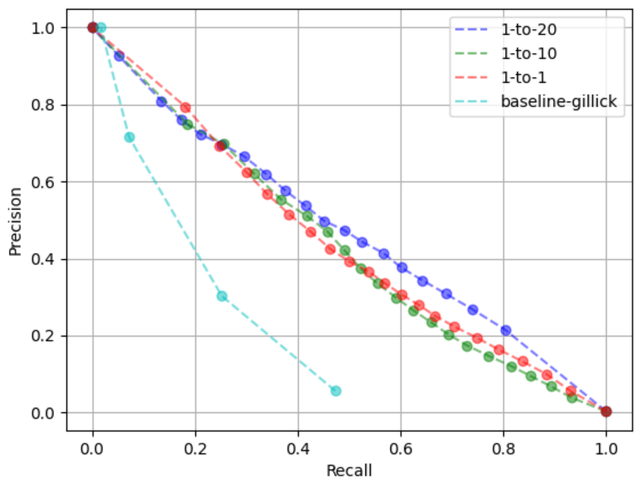
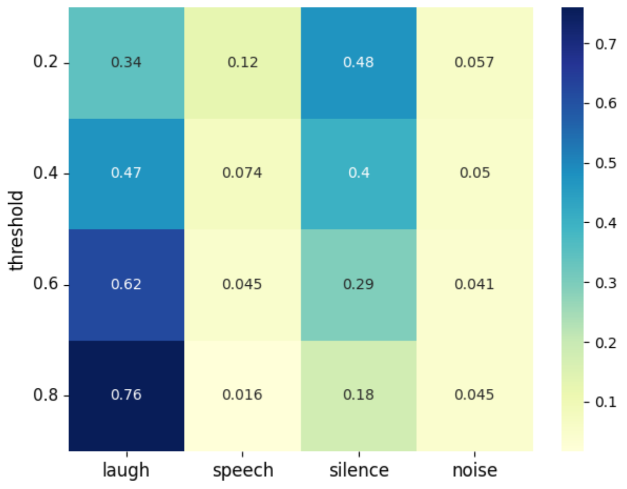


Figure 5.1: Precision-Recall-Curve for different ratios between laughter and non-laughter based on 21 evenly spaced thresholds from 0 to 1. The baseline, given by the model from Gillick et al. (2021) evaluated the development set, only has four data points for thresholds [0.2,0.4,0.6,0.8], because it uses the initial evaluation from Chapter 3.

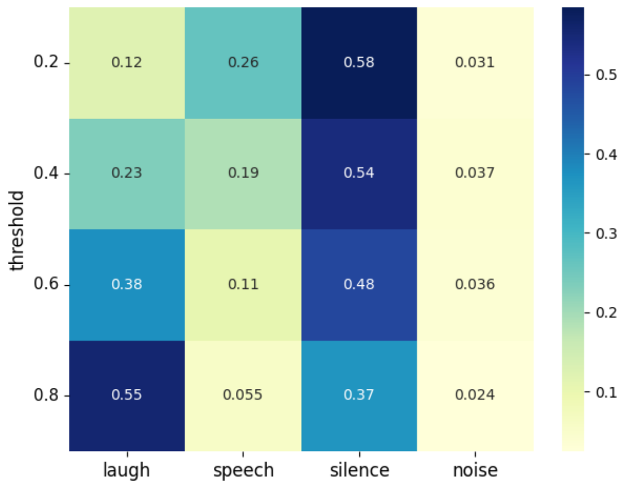
baseline is not the same as Figure 3.2, because this time we only evaluate on the development set. For all newly trained models, most misclassifications are in silence regions, regions that have no transcription. In contrast, the baseline classifies more laughter samples as speech than silence. Generally, we note that there is only a small amount of samples located in noise regions, the baseline has almost none. The 1-to-20 model has the least number of speech samples across all thresholds, except for threshold 0.8 where the baseline has 100% precision, i.e., all the predicted laughter samples are indeed laughter samples.

5.1.3 Analysis

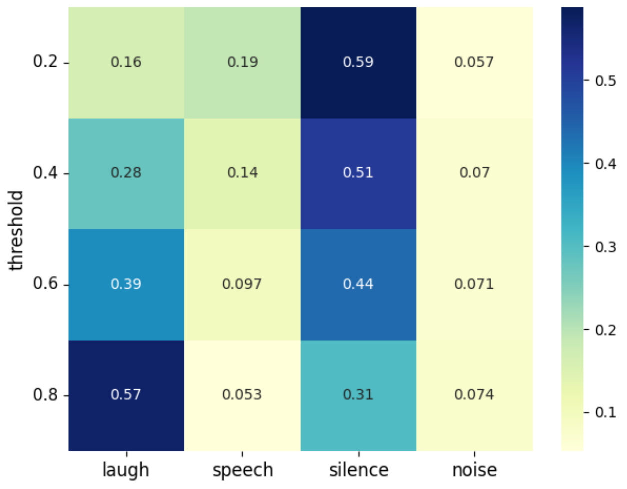
The purpose of the experiment was to investigate different class balances. The precision-recall curve (Figure 5.3) suggests that the 1-to-20 ratio gives the best performance. Nevertheless, for integrating such a model into a video call system, the performance is still poor. The three models trained on the ICSI corpus perform significantly better than the baseline of Gillick et al. (2021) that was trained on the Switchboard dataset (as in Chapter 3). This behaviour is expected and aligns with findings from Gillick et al. (2021) (Figure 2.3). A model evaluated on the same dataset it was trained on performs



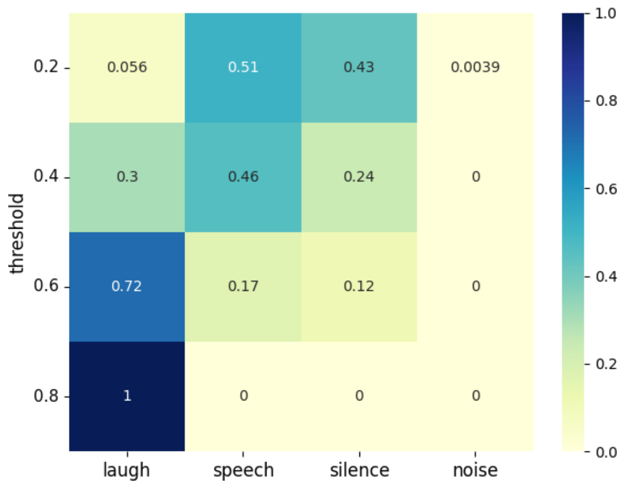
(a) 1-to-20



(b) 1-to-10



(c) 1-to-1



(d) baseline-gillick

Figure 5.2: Confusion matrices for different ratios between laughter and non-laughter. Each row shows the percentage of laughter segments classified as a certain subclass for a given threshold (see Section 2.3.2). Baseline is given by the model from Gillick et al. (2021) evaluated on the development set (note: Figure 3.2 was evaluated on the whole ICSI corpus).

better than a model trained on a different dataset.

Another insight is deduced from the confusion matrices. Whereas the baseline has more misclassified speech than silence segments, the three newly trained models misclassify speech less often. In contrast, the newly trained models often misclassify laughter as silence, regions with no transcription. Evaluating the pre-trained model by

Gillick et al. (2021) on the whole ICSI corpus (Chapter 3) showed a non-negligible amount of misclassification in silence regions. These newly trained models make this problem even more apparent and raise the question of why the model is unable to confidently separate laughter from silence. Section 5.3 will investigate this further.

Lastly, the pre-trained model's 100% precision for a threshold of 0.8 does not necessarily indicate a good model performance, as explained in Section 2.3.1. Looking at Figure 5.1, this threshold corresponds to the light blue data point in the upper left corner. This shows that this threshold is close to one extreme of the precision recall curve where precision is 100% and recall is 0%. Future work could investigate why the pre-trained model already reaches this extreme at a threshold of 0.8, but compared to other future tasks, I consider this as less important.

5.2 Experiment 2 - Non-laughter Diversity

The second set of experiments investigates the impact of the amount of different types of non-laughter on the model performance. The ratio of non-laughter segments is fixed to 1-to-20. The ratio 1-to-20 is chosen to maximise the amount of non-laughter segments and be able to use one of models from Section 5.1 as a baseline. The segment selection and feature computation process is similar to Section 5.1, but non-laughter segments were not chosen at random. A fixed percentage of non-laughter segments was assigned to each of the three types of non-laughter: speech, noise, and silence. Table 5.1 shows the different percentages tried.

model	speech	silence	noise
a	70%	10%	20%
b	10%	70%	20%
c	20%	70%	10%

Table 5.1: Percentages of non-laughter subclasses for experiment 2.

5.2.1 Results

Looking at the precision-recall curve in Figure 5.3, we see that `model-a` performs worse than the baseline across all thresholds. `model-c` performs similar to the baseline with worse performance in the area of `recall` > 0.6. `model-b` performs similar to

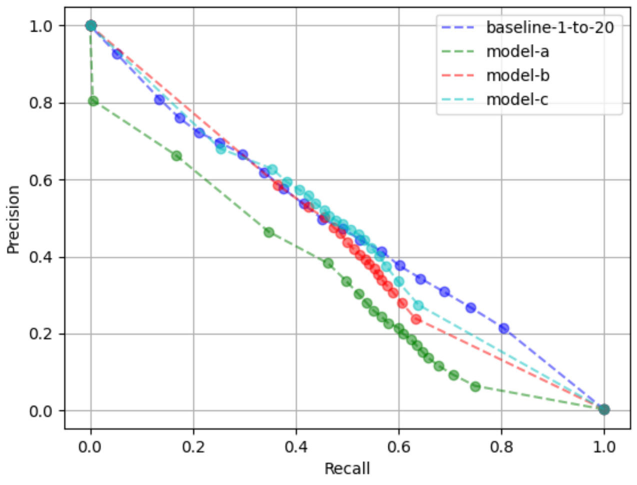


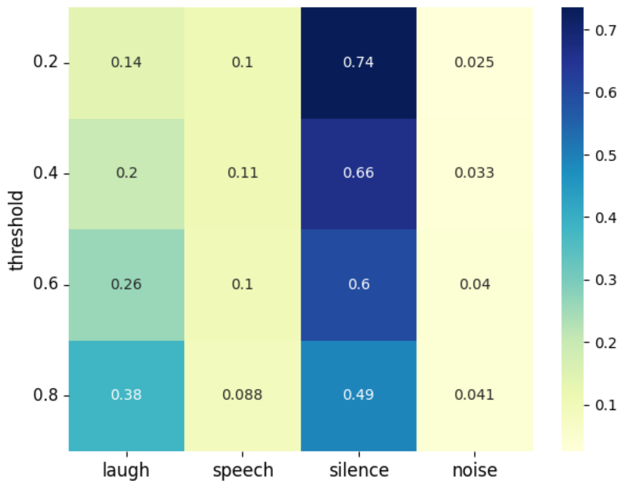
Figure 5.3: Precision-Recall-Curve for different splits of non-laughter subclasses as shown in Table 5.1. Baseline taken from experiment one (Section 5.1) where non-laughter segments were sampled randomly.

`model-c`, but slightly worse. We note that the majority of data points for `model-b` and `model-c` are condensed in the area of $0.3 < \texttt{recall} < 0.7$ and $0.2 < \texttt{precision} < 0.65$. For `model-c` most data points lie in the area of $0.4 < \texttt{recall} < 0.75$ and $0.1 < \texttt{precision} < 0.4$.

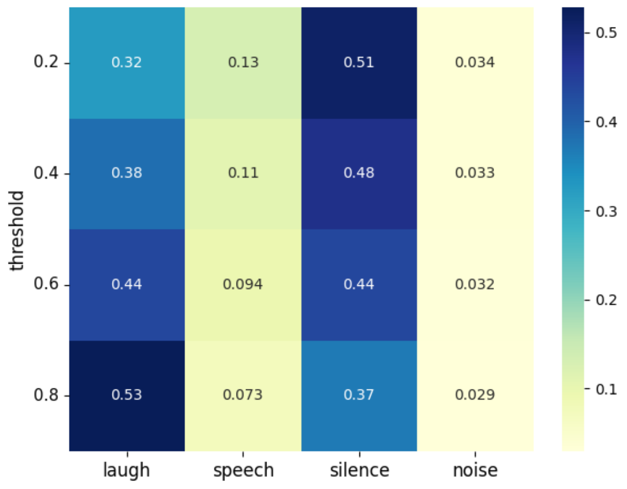
Figure 5.4 compares the confusion matrices. The general structure is similar to the matrices in Figure 5.2. Most of the misclassifications are silence segments. The small number of data points in higher precision regions of the precision-recall curve (Figure 5.3) becomes evident in the matrices as well. As described in Section 2.3.2, the laughter percentage in the confusion matrix is equal to the precision. Thus, of the newly trained models for a threshold of 0.8 is low. Lastly, we can see that `model-c` classifies more laughter segments as silence than as laughter.

5.2.2 Analysis

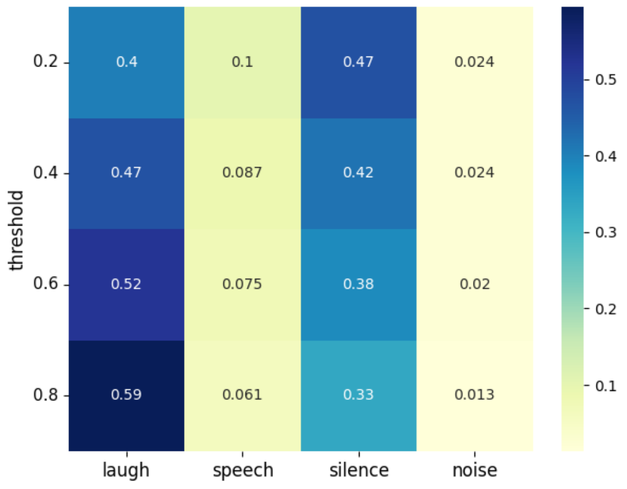
The goal of these experiments was to analyse the impact of diversity within the non-laughter segments. As shown by `model-a`, using a small number of silence segments significantly decreases model performance. It causes the model to classify more silence segments as laughter than true laughter segments. This is a major flaw in this model and suggests that this subset of the ICSI corpus is not representative of its entirety.



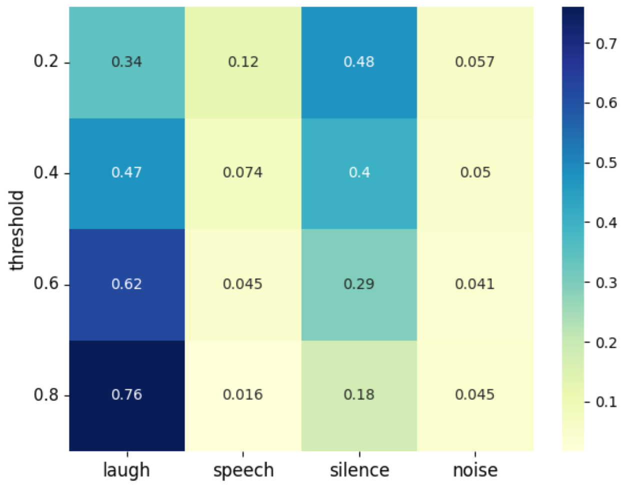
(a) model-a: 70sp, 10sil, 20noi



(b) model-b: 10sp, 10sil, 20noi



(c) model-c: 20sp, 70sil, 10noi



(d) baseline: 1-to-20 model from Section 5.1

Figure 5.4: Confusion matrices for different splits of non-laughter types as shown in Table 5.1. Each row shows the percentage of laughter segments classified as a certain subclass for a given threshold (see Section 2.3.2). Splits shown as percentages, where sp=speech, sil=silence, noi=noise.

Keeping the number of silence segments constant and slightly increasing the number of speech segments yielded a small performance impact, as shown by `model-b` and `model-c`.

Thus, for this experiment, changing the number of silence segments in the non-laughter class has the biggest discriminative power. This is surprising, because intuitively one expects that silence should not have a lot of variety and thus, increasing the number of training samples should not have a big impact. Possible reasons for this are investigated in the next section.

Lastly, the low precision values for a threshold of 0.8 compared to the baseline, do not mean that the new models in this experiment perform worse. As mentioned before, thresholds are variable and we should always consider the whole precision-recall curve. Nevertheless, it is interesting to note that changing the type of non-laughter segments seems to impact the way thresholds partition the model performance. This fact should be investigated further in future work.

5.3 Investigating Non-transcribed Regions in the ICSI Corpus

Both, the initial evaluation (Chapter 3) and the experiments (Chapter 5) raised the question, why the model is not able to separate laughter from silence segments in the corpus. Thus, I decided to manually analyse some audio tracks for abnormalities. This is a crude analysis to come up with a hypothesis for the large amount of retrieved silence segments.

I randomly selected five of the 468 audio tracks and plotted their audio-wave (Figure 5.5). The selected tracks are plotted in this order:

1. Bmr027: chan5
2. Bdb001: chan4
3. Btr001: chan7
4. Bns003: chan0
5. Bmr018: chan1

Depending on the activity and looking at the corresponding transcript I listened to segments of the recordings that were supposed to be silent. In all cases, except for `Bmr027:chan5`, I was able to clearly hear the other participants speaking. `Bmr027:chan5` (the top recording in Figure 5.5) was recorded with a different headset than the other four tracks which is likely the reason why there is actual silence on this track. `Bns003:chan0` (fourth row in Figure 5.5) is the most problematic. There is almost no difference between the volume of the participant speaking to the rest of the meeting. Figure 5.6

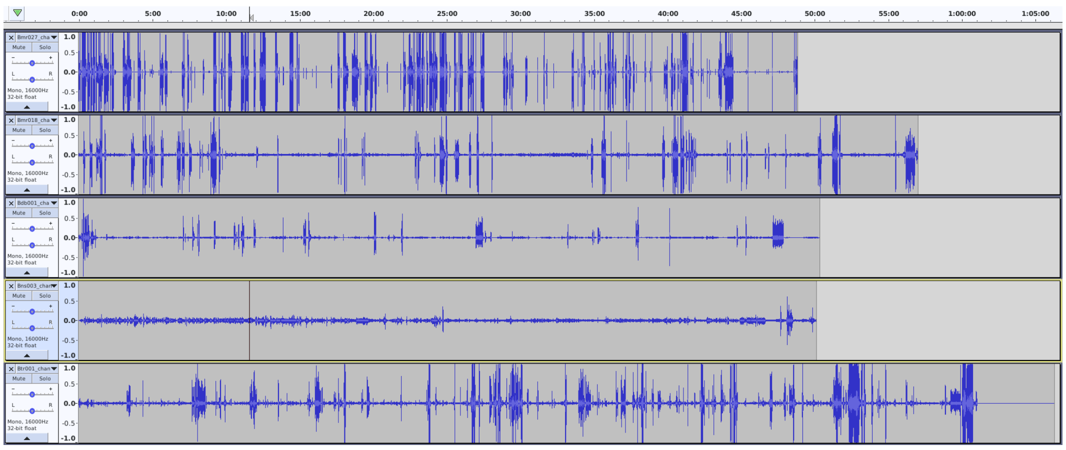


Figure 5.5: Audio waves of five random audio tracks from the ICSI corpus.

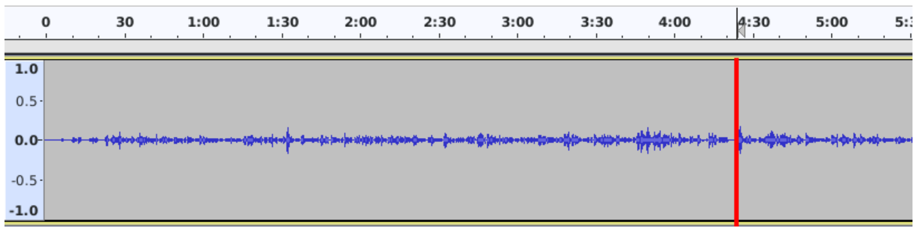


Figure 5.6: `Bns001:chan0`. First transcribed occurrence marked by red line. Considering transcriptions, everything else should be silence. This is not visible in the audio wave.

shows the first transcribed occurrence, a short laughter, which is not distinguishable from the rest of the audio wave.

From this analysis, I conclude the falsity of the assumption that everything not transcribed is silence. Thus, the areas classified as silence in the preprocessing (3.2) are not actually silence for a significant number of cases; they are quieter recordings of speech by other participants. This has two immediate consequences. Firstly, to get a representative subset of the ICSI corpus significant amounts of non-transcribed regions need to be sampled, as shown in Section 5.2. Secondly, the evaluation code needs to be adapted because the current version (Wolter, 2022a) only counts laughter occurrences in transcribed regions as true positives. If a laughter event occurs within a non-transcribed region, and is correctly predicted as such, it will not be considered a correct prediction. In hindsight, such an analysis at an earlier stage of the project would have been beneficial and prevented false assumptions.

Future work should further investigate the non-transcribed regions and explore methods to deal with them. One approach could be to examine all channels recorded with the headset type used by channel `Bmr027:chan5`, tagged with `c1` in the ICSI transcriptions (Morgan et al., 2001; ICSI, n.d.). If all of these recordings proof to be of higher quality in terms of separating activity from silence, only evaluating on those channels could reveal if noise within non-transcribed regions is the issue.

Another approach would be to filter for meetings with little volume variation across the whole duration (like `Bns003:chan0`). Such meetings could be excluded from the evaluation. Alternatively, a minimum volume filter could be applied to disregard all audio activity below a certain threshold. Finding a good threshold would be essential for such a filter. With a low threshold, the filter might not make a difference, a large threshold, on the other hand, might disregard lots of quieter laughter occurrences. Other approaches are possible and should be investigated in future work.

Chapter 6

Practicality of Laughter Recognition in Video Meetings

This section covers the practicality of a `filtered` option within a video call system. As mentioned in Chapter 1 such an option would exist alongside `mute` and `unmute` and automatically unmutes a user when laughter is detected. This section is split into three parts:

1. Real-time considerations for such a system
2. Practicality and privacy issues
3. Alternative solutions to the approach presented in this thesis

6.1 Laughter Detection in Real-time

Due to time constraints, this chapter contains only theoretical considerations and pointers for future work. For detecting laughter in video meetings, the model needs to work in real-time and with low computational cost. Real-time here means low latency. High retrieval accuracy becomes useless if the detected laughter is only fed back to the audio stream after a few seconds.

There are six factors that influence the model's latency:

1. architecture: client vs. server side implementation
2. programming language used

3. computational power available
4. model's complexity
5. model's window-size
6. minimum length of retrieved laughter segment

A detection algorithm could be implemented on either the client side, e.g. using JavaScript, or on the server side, e.g. using Python. This design decision influences the programming language used and the computational power available to the detection algorithm. If the detection runs on the client side, the computational power is limited by the user's end-device. On the contrary, the computational power on the server side is almost unlimited. However, a low computational footprint is desirable because server costs grow with increasing computational power. Since this project does not cover the integration within a video call system, the discussion about programming languages and architectural decisions between client- and server-side is left for future work.

Assuming a fixed amount of computational power, reducing the model's complexity improves the computational efficiency and latency. Gillick et al.'s (2021) model used in this thesis, a ResNet architecture with 18 layers, is not computationally efficient. Existing research investigates the reduction of the computational footprint with minimal impact on performance (Meyer et al., 2017; Sørensen et al., 2020). Future work should review such literature and see how it can be applied to this project.

The feature representation is another important design decision that affects the latency. All newly trained models in Chapter 5 use mel spectrograms of shape 100x40, which means 100 samples and 40 filters. Each feature represents a one-second segment, and thus, the window size is one second. Splitting this window evenly around the target frame, similar to Figure 2.1, we get around 0.5s before and 0.5s after the considered frame. For real-time processing, this causes a minimum latency of 0.5s, which is unacceptable for our system. There are two possible solutions, that should be investigated in future work: reducing the window size, which causes loss of context information, or using an uneven split, e.g. three-fourths before the frame and one-fourth after.

Lastly, the minimum length is an inference parameter used by the model from Gillick et al. (2021). Even though this parameter prevents the retrieval of laughter occurrences that might be considered too short, this is done at the cost of latency. The model needs to wait at least the specified amount of time before a laughter occurrence is approved.

Thus, the minimum latency is the duration of the minimum length itself. Future work should investigate whether a minimum-length parameter is needed at all or if smarter use of the context window can avoid this additional parameter. In summary, assuming a minimum length m and an evenly split window of size w, a lower bound of the model's latency is given by: $m + \frac{1}{2}w$.

	i5-6500 CPU @ 3.20GHz		i5-8500 CPU @ 3.00GHz		GeForce GTX 1060 6GB	
audio duration	iterations	av-RTF	iterations	av-RTF	iterations	av-RTF
3s	20	1.31	20	0.63	20	0.14
30s	20	1.41	20	0.84	20	0.10
120s	10	1.49	10	0.81	20	0.10

Table 6.1: Average real-time factor for inference on different machines (2x CPU, 1xGPU).

Table 6.1 provides a baseline for future work that estimates the latency of the model from Gillick et al. (2021) using a window size of one second. The models trained in Chapter 5 will have a similar latency because they use the same ResNet architecture and the same window size. The model evaluated uses a window size of one second. The latency is estimated by the real-time factor(RTF), which is calculated as:

$$RTF = \frac{time\ to\ process\ a\ segment}{duration\ of\ the\ segment}$$

Note that this is an optimistic estimate, as the RTF is calculated for longer segments. Since the minimum length is a postfilter applied to the model output, the overhead caused by minimum length parameter for a real-time model described in Section 6.1 does not apply for longer segments.

6.2 Practicality Issues and Privacy

Even though users can manually enable the `filtered` option during a video call, it is important to think about privacy. If most users do not feel comfortable using this option, it will significantly affect its practicality. The audio of every meeting participant will be constantly analysed. Even if the system promised to discard all audio immediately, users might feel uncomfortable using such a system. False positives in particular are a problem. If the system detects something as laughter and feeds it into the audio stream, it is irreversible. If the false positive is a sensitive conversation with a friend, the cost of this issue is huge. The extent of this issue can be limited by tuning the model threshold in such a way that the false positive rate is minimised. Nevertheless, the use case outlined above will always remain possible.

The architecture of the system also influences the practicality. Whereas, a client-side implementation will analyse the data locally, a server-side implementation implies processing sensitive audio data on the server. On the one hand, people might mistrust the implementation and doubt that data is deleted from the server after analysis. Furthermore, sending the data to the server requires the transmission of sensitive audio data over public networks, which opens up an attack surface for bad actors.

Another general problem is the missing detection of laughter sentiment. Laughter in response to a joke will likely please the speaker. In contrast, laughter that makes fun of the speaker might seriously affect the speaker's confidence. I talked to a member of university staff who told me that she would be afraid of using such a system for this reason. Using a machine learning model to pick up the sentiment conveyed by a laughter occurrence is a completely different task. Even if this did succeed to a certain degree, false positives would still be an issue.

6.3 Alternative Solutions

Considering the privacy concerns mentioned in Section 6.2, we can think about alternative sensory data. Constantly analysing video input will likely cause similar discomfort for participants. Additionally, video input is often turned off during larger video conferences. Cosentino et al. (2016) also mention the privacy implications of video and audio inputs and suggest the use of individual wearable sensors to address them. For our project, this approach was not feasible due to the lack of available sensory data. It is also impractical to deploy such a system on a large scale. Participants join video conferences from everywhere around the world, where wearable sensors are not available to them. Lastly, it is debatable whether a wearable sensor brings more comfort to participants than constantly being listened to.

During a presentation, a student suggested an interesting idea. He suggested constantly analysing all audio inputs of the meeting without looking for laughter specifically. Considering that usually one person is speaking, audio activity from lots of inputs synchronously suggests some common activity, e.g., laughter or applause. Trying to identify patterns like this is an interesting alternative that is worth considering.

Chapter 7

Conclusion and Future Work

The original objective of the project was to create a filter for laughter and applause and integrate it as a new `filtered` option, to complement the `mute` and `unmute` options in video call systems. This goal was not achieved, but the project provides the foundation for the laughter recognition component of such a system. Although the current performance is not sufficient, the insights on training such a model on the ICSI corpus (Morgan et al., 2001), my final hypothesis, and the published code (Wolter, 2022a) will benefit future work.

Prior research (Kennedy and Ellis, 2004; Knox, 2006; Truong and Leeuwen, 2005) only used parts of the ICSI corpus for training and evaluation and discarded non-transcribed regions during preprocessing. This thesis makes use of the whole ICSI corpus and includes non-transcribed regions in the evaluation. Since laughter recognition is a small research area, with little published code available, the insights and the published code (Wolter, 2022a) from this thesis can be valuable for laughter recognition research in general and especially lower the barrier for novices in the field like myself at the beginning of the project. However, it should be noted that some parts of the code, especially `train.py`, still contain large parts of the code of Gillick et al. (2021) that I criticised as cumbersome and not adhering to common PyTorch standards.

My initial evaluation in Chapter 3 and the experiments conducted in Chapter 5 resulted in my final hypothesis:

"A significant number of non-transcribed regions in the ICSI corpus are not actually silence; they are quieter recordings of speech by other participants. Thus, the assumption that non-transcribed regions and laughter can be easily distinguished is wrong."

Following prior research and disregarding the non-transcribed regions completely is a possible solution, but not ideal for our use case. Using continuous audio tracks instead of disconnected laughter and non-laughter samples better resembles the use case of laughter detection as part of a video call system. Therefore, future work should investigate how to address this issue without discarding non-transcribed regions entirely. Section 5.3 proposes two approaches: only using tracks that actually contain silence or applying more advanced preprocessing.

Future research can also use the published code (Wolter, 2022a) to continue the experiments presented in Chapter 5. Conducting a grid search on the variable parameters in those experiments could determine the corpus subset that is most representative of the whole ICSI corpus. Adapting the feature representation could be another parameter to experiment with. The thesis mentions three ideas for this: different spectrogram dimensions, feature augmentation, and subsampling.

Making the model work in real-time is another important part for the continuation of this project. Section 6.1 provides some theoretical considerations as well as a baseline for future work. Lastly, ethical considerations like user privacy and emotional harm are only covered superficially in Chapter 6. For a system that continually processes audio from end users, a more thorough ethical review is vital.

Bibliography

Lhotse: a speech data representation library for the modern deep learning ecosystem, NeurIPS Data-Centric AI Workshop, 2021. URL https://arxiv.org/abs/2110.12561.

Corine A Bickley and Sheri Hunnicutt. Acoustic analysis of laughter. In *ICSLP*, volume 51, pages 52–55, 1992.

Rui Cai, Lie Lu, Hong-Jiang Zhang, and Lian-Hong Cai. Highlight sound effects detection in audio stream. In *2003 International Conference on Multimedia and Expo. ICME'03. Proceedings (Cat. No. 03TH8698)*, volume 3, pages III–37. IEEE, 2003.

Sarah Cosentino, Salvatore Sessa, and Atsuo Takanishi. Quantitative laughter detection, measurement, and classification—a critical survey. *IEEE Reviews in Biomedical engineering*, 9:148–162, 2016.

John Garofolo, Christophe Laprun, and Jonathan Fiscus. The rich transcription 2004 spring meeting recognition evaluation, 2004-04-01 2004.

Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

Jon Gillick, Wesley Deng, Kimiko Ryokai, and David Bamman. Robust laughter detection in noisy environments. *Proc. Interspeech 2021*, pages 2481–2485, 2021.

J.J. Godfrey, E.C. Holliman, and J. McDaniel. Switchboard: telephone speech corpus for research and development. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520 vol.1, 1992. doi: 10.1109/ICASSP.1992.225858.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

ICSI. Meeting recorder naming conventions, n.d. URL `https://www1.icsi.berkeley.edu/Speech/mr/naming.html`.

Jrgillick. Jrgillick/laughter-detection, 2021. URL `https://github.com/jrgillick/laughter-detection`.

Lyndon S Kennedy and Daniel PW Ellis. Laughter detection in meetings. 2004.

Mary Knox. Automatic laughter detection. *Final Project (EECS 294)*, 2006.

LDC. Icsi meeting speech, n.d.a. URL `https://catalog.ldc.upenn.edu/LDC2004S02`.

LDC. Switchboard-1 release 2, n.d.b. URL `https://catalog.ldc.upenn.edu/LDC97S62`.

Yan-Xiong Li, Qian-Hua He, Sam Kwong, Tao Li, and Ji-Chen Yang. Characteristics-based effective applause detection for meeting speech. *Signal Processing*, 89(8): 1625–1633, 2009.

C Manoj, S Magesh, Aditya Sriram Sankaran, and M Sabarimalai Manikandan. Novel approach for detecting applause in continuous meeting speech. In *2011 3rd International Conference on Electronics Computer Technology*, volume 3, pages 182–186. IEEE, 2011.

Brian McFee, Colin Raffel, Dawen Liang, Daniel P Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*, volume 8, pages 18–25. Citeseer, 2015.

Annamaria Mesaros, Toni Heittola, and Tuomas Virtanen. Metrics for polyphonic sound event detection. *Applied Sciences*, 6(6):162, 2016.

Matthias Meyer, Lukas Cavigelli, and Lothar Thiele. Efficient convolutional neural network for audio event detection. *arXiv preprint arXiv:1709.09888*, 2017.

Nelson Morgan, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Adam Janin, Thilo Pfau, Elizabeth Shriberg, and Andreas Stolcke. The meeting project at icsi.

In *Proceedings of the first international conference on human language technology research*, 2001.

Nelleke Oostdijk et al. The spoken dutch corpus. overview and first evaluation. In *LREC*, pages 887–894. Athens, Greece, 2000.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Jeff Reback, jbrockmendel, Wes McKinney, Joris Van den Bossche, Tom Augspurger, Phillip Cloud, Simon Hawkins, gfyoung, Matthew Roeschke, Sinhrks, Adam Klein, Terji Petersen, Jeff Tratner, Chang She, William Ayd, Patrick Hoefler, Shahar Naveh, Marc Garcia, Jeremy Schendel, Andy Hayden, Daniel Saxton, JHM Darbyshire, Richard Shadrach, Marco Edward Gorelli, Fangchen Li, Matthew Zeitlin, Vytautas Jancauskas, Ali McMaster, Pietro Battiston, and Skipper Seabold. pandas-dev/pandas: Pandas 1.3.4, October 2021. URL https://doi.org/10.5281/zenodo.5574486.

Steve Renals and Pawel Swietojanski. Neural networks for distant speech recognition. In *2014 4th joint workshop on hands-free speech communication and microphone arrays (HSCMA)*, pages 172–176. IEEE, 2014.

Kimiko Ryokai, Elena Duran Lopez, Noura Howell, Jon Gillick, and David Bamman. Capturing, representing, and interacting with laughter. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018.

Peter Mølgaard Sørensen, Bastian Epp, and Tobias May. A depthwise separable convolutional neural network for keyword spotting on an embedded system. *EURASIP Journal on Audio, Speech, and Music Processing*, 2020(1):1–14, 2020.

Khiet P Truong and David A van Leeuwen. Automatic detection of laughter. In *Ninth European Conference on Speech Communication and Technology*, 2005.

Khiet P Truong and David A Van Leeuwen. Automatic discrimination between laughter and speech. *Speech Communication*, 49(2):144–158, 2007.

Christian Uhle. Applause sound detection. *Journal of the Audio Engineering Society*, 59(4):213–224, 2011.

Lasse Wolter. A Machine Learning Pipeline for Laughter Detection on the ICSI Corpus, 4 2022a. URL `https://github.com/LasseWolter/laughter-detection-icsi`.

Lasse Wolter. Honours Project 2021/22 - A Machine Learning Model for Laughter Detection, 4 2022b. URL `https://github.com/LasseWolter/honours-project-21-22`.

Appendix A

First appendix

A.1 Lhotse Contributions

Pull Requests in chronological order

1. minor documentation fix in *ICSI-recipe*: PR 1
2. added missing microphone channels to *ICSI-recipe*: PR 2
3. added warning for certain case of feature computation: PR 3
4. improved *ICSI-recipe* download structure: PR 4 and PR 5

A.2 Binary Cross Entropy Loss

In comparison to accuracy, precision, and recall (Section 2.3.1) the loss of a model is computed from the raw probabilities output by the model, not the predicted labels.

The machine learning model outputs a probability between 0 and 1. Only applying a threshold turns this probability into a class label. This threshold is a variable parameter. For example, with a threshold of 0.5, all predicted probabilities of 0.5 and higher are output as class 1, in our case laughter. All probabilities below 0.5 are output as non-laughter. Varying this threshold changes the predicted labels and thus, yields a different performance in terms of precision and recall.

In contrast, the loss of a model's output is independent of the threshold as it is calculated directly from the probabilities. When a machine learning algorithm learns how to improve its predictions, it minimises this loss. There are different functions for calculating loss. A typical approach for classification problems is the Cross Entropy Loss. Since our use case only has two classes (laughter and non-laughter) we can use Binary Cross Entropy Loss.

Assume we have two vectors P and Y of length m. For each of the m samples, P contains the probability of the sample being laughter whereas Y contains its true class (laughter $\Rightarrow y = 1$, non-laughter $\Rightarrow y = 0$). Given this, the model's Binary Cross Entropy Loss is given by:

$$L(P,Y) = \sum_{n=1}^{m} -(y * \log(p) + (1-y) * \log(1-p))$$

To understand the meaning of this equation, we look at a single sample with predicted probability p and a true class value of y. Its Binary Entropy Loss is given by:

$$L(p,y) = -(y * \log(p) + (1-y) * \log(1-p))$$

We know that one of the terms y and $(1-y)$ will always be zero. Thus, we are left with the negative log of the probability that the sample belongs to the true class, e.g., a sample with $p = 0.8$ is predicted to be laughter with an 80% chance and non-laughter with a 20% chance. Negative log ensures that $L(p,y) > 1$ for all p. Due to the logarithmic nature of this loss-function confidently misclassified examples (e.g., $p = 0.1$) are punished more heavily than unconfidently misclassified examples (e.g. $p = 0.4$).

To summarise, the Binary Cross Entropy Loss captures how far off the predictions of the model are from the true values. It sums up the negative log probabilities of belonging to the true class and punishes confidently misclassified samples more heavily.