Computer Speech and Language 81 (2023) 101519

Contents lists available at ScienceDirect

Computer Speech & Language

journal homepage: www.elsevier.com/locate/csl

Classification of stuttering – The ComParE challenge and beyond

Sebastian P. Bayerl [a,*], Maurice Gerczuk [c], Anton Batliner [c], Christian Bergler [b], Shahin Amiriparian [c], Björn Schuller [c,d], Elmar Nöth [b], Korbinian Riedhammer [a]

[a] Technische Hochschule Nürnberg, Nürnberg, 90489, Germany
[b] Friedrich-Alexander Universität Erlangen-Nürnberg, Erlangen, 91058, Germany
[c] Universität Augsburg, Chair for Embedded Intelligence for Health Care and Wellbeing, Augsburg, 86159, Germany
[d] Imperial College London, GLAM – Group on Language, Audio, & Music, London, SW7 2AZ, UK

ARTICLE INFO

Keywords:
Dysfluency
Stuttering
ComParE challenge
Paralinguistics
Pathological speech

ABSTRACT

The ACM Multimedia 2022 Computational Paralinguistics Challenge (ComParE) featured a sub-challenge on the classification of stuttering in order to bring attention to this important topic and engage a wider research community. Stuttering is a complex speech disorder characterized by blocks, prolongations of sounds and syllables, and repetitions of sounds and words. Accurately classifying the symptoms of stuttering has implications for the development of self-help tools and specialized automatic speech recognition systems (ASR) that can handle atypical speech patterns. This paper provides a review of the challenge contributions and improves upon them with new state-of-the-art classification results for the KSF-C dataset, and explores cross-language training to demonstrate the potential of datasets in multiple languages. To facilitate further research and reproducibility, the full KSF-C dataset, including test-set labels, is also released.

1. Introduction

Disfluencies naturally occur in speech. Especially conversations and everyday language have more disfluencies than formal speech. The amount of disfluencies that occur in a person's speech is influenced by factors such as the current cognitive load, psychological factors, or their state of sleepiness. These disfluencies can be interjections which are typically used to gain time to think while speaking (Belz, 2021), or might be caused by an attempt to repair or revise the original sentence by restarting or rephrasing the sentence.

In the context of this paper, we follow the definition of dysfluencies by Ward et al. (2013, p. 95). Dysfluency means abnormality of fluency, as indicated by the Greek stemming prefix 'dys-', which means abnormal, impaired, or bad.[1] This is different from the prefix 'dis' which means the opposite of or simply 'not'.[2] Dysfluency includes, but is not limited to, stuttering (Wingate, 1984). The speech patterns associated with dysfluencies are diverse, including aphasia, dysarthria, and apraxia. Dysfluencies, as opposed to disfluencies, have a pathological cause or occur unusually often. Causes for dysfluent speech might be neurological anomalies, physical impairment, brain damage, or hearing loss. In the case of stuttering, the prevalent cause that researchers widely accept are neurological anomalies with a hereditary component (Yairi and Ambrose, 2013), affecting more biological males than females (Craig et al., 2002; Sommer et al., 2021).

* Correspondence to: Technische Hochschule Nürnberg Georg Simon Ohm, Keßlerplatz 12, 90489, Nuremberg, Germany.
E-mail address: sebastian.bayerl@th-nuernberg.de (S.P. Bayerl).

[1] https://www.merriam-webster.com/dictionary/dys-
[2] https://www.merriam-webster.com/dictionary/dis-

https://doi.org/10.1016/j.csl.2023.101519
Received 2 January 2023; Received in revised form 7 February 2023; Accepted 28 March 2023
Available online 3 April 2023

Stuttering can be identified by an unusual amount of functional disfluencies as well as blocks, prolongations of sounds and syllables, and repetitions of sounds and words (Ellis and Ramig, 2009). These so-called core symptoms are accompanied by diverse linguistic, physical, behavioral, and emotional symptoms. Stuttering is a variable condition. The characteristics of stuttering symptoms and severity vary substantially between speakers and for a person who stutters (PWS) depending on various factors. The communication situation, psychological factors, and the linguistic complexity of an utterance influence the appearance of stuttering symptoms (Ellis and Ramig, 2009). These symptoms hinder PWSs from communicating freely and thus negatively influence different aspects of the life of affected people. The variability of this fluency disorder makes it hard to detect. Thus resources for training detection systems can hardly capture all aspects of stuttering. Detecting and classifying stuttering has implications for developing inclusive voice technology, ranging from specially adapted automatic speech recognition systems (ASR) to self-help applications. The results and insights from detection and classification experiments are also transferable to other paralinguistic phenomena, such as apraxia or Parkinson's disease, where people might be dysfluent because of reduced motor control over their primary articulators.

Machine learning for stuttering classification has a long history in pathological speech processing (Howell and Sackin, 1995; Howell et al., 2009; Noeth et al., 2000). The topic has gained momentum in recent years as evidenced by the works of Kourkounakis et al. (2020, 2021), Kourkounakis (2021), Bayerl et al. (2020, 2022,b,c), Lea et al. (2021), and Sheikh et al. (2021, 2022a).

One of the latest contributions to machine learning for stuttering detection and classifications was the ACM Multimedia 2022 computational paralinguistics challenge (ComParE) that featured a sub-task on stuttering classification (Schuller et al., 2022). This paper contributes a comprehensive review of the contributions received for the stuttering sub-challenge of the 2022 ComParE challenge, identifies trends, and formulates current challenges in the field. Additionally, we describe a wav2vec 2.0-based method and compare it to the challenge contributions. To facilitate the reproducibility of methods and further research in stuttering classification, we publish an updated version of the Kassel State of Fluency challenge (KSF-C) dataset that includes the test-set labels.[3]

This paper is organized as follows; Section 2 briefly describes the data used in the challenge and this paper. Section 3 section gives an overview of the shared methods by the challenge participants and a description of methods used in the experiments in this paper. Section 4 describes related work w.r.t. machine learning for stuttering and contains a detailed overview of the challenge contributions. Section 5 contains a description of the experiments performed within the scope of this paper followed by Section 6 discussion of the results, a comparison of the challenge contributions, and a description of open problems and challenges in stuttering classification.

2. Data

2.1. Kassel State of Fluency

The Kassel State of Fluency Challenge (KSF-C) corpus is a derived, simplified version of the Kassel State of Fluency (KSoF) corpus created by Nuremberg Tech (Technische Hochschule Nürnberg) and recorded at the Kasseler Stottertherapie (Bayerl et al., 2022). The original corpus consists of 5597 typical and atypical (stuttering) 3 s long speech segments from 37 German speakers with a total duration of 4.6 h. All segments were annotated by three naive labelers as belonging to one of seven stuttering-related classes (block, prolongation, sound repetition, word/phrase repetition, modified speech technique, interjection, no disfluency) (Bayerl et al., 2022). The annotations are complemented by meta-labels, e.g., about the recording quality or the presence of background noise. The distribution of labels and meta-labels from the original dataset can be taken from Table 1 The percentage of stuttering labels sums up to > 100 % because annotators could assign more than one stuttering and non-stuttering label to each clip. For the ComParE challenge, the organizers removed all segments labeled with more than one stuttering-related label. Thus, only features 4601 segments with a unique label. The challenge tasks consist of the classification of speech segments as one of eight classes — the seven stuttering-related classes and an eighth 'garbage' class, denoting segments that are unintelligible, contain no speech, or are negatively affected by background noise. The dataset was split into three speaker-independent partitions (Train: 23 speakers, Dev: 6 speakers, Test: 8 speakers) and label distribution can be taken from Table 2.

The challenge data, including the test set labels, will be made available through zenodo.[4]

2.2. Stuttering Events in Podcasts

For our own experiments, we use two additional corpora containing 3-second long clips with stuttered speech; SEP-28k-Extended and FluencyBank (Bayerl et al., 2022c; Lea et al., 2021; Bernstein Ratner and MacWhinney, 2018). The Stuttering Events in Podcasts extended (SEP-28k-E) dataset is based on the SEP-28k corpus (Lea et al., 2021). The extension comes with automatically generated speaker labels and a speaker-exclusive Train-Dev-Test split to enable reproducibility of results.[5]

The original release of the SEP-28k corpus contains 4144 English clips extracted from the Voices of Adults Who Stutter part data from the FluencyBank corpus (Bernstein Ratner and MacWhinney, 2018). The clips were annotated using the same labeling protocol as SEP-28k and the labels are compatible with the labels in the KSoF corpus. The clips in SEP-28k and FluencyBank were annotated with similar meta information to KSoF, such as music, background noise, and poor audio quality. The three datasets have compatible labels for the five core-symptom; blocks, prolongations, sound repetitions, word repetitions, and interjections.

[3] KSF-C available upon request from https://zenodo.org/record/6460102

[4] EULA and instructions to get the data https://tinyurl.com/4f9hyny2

[5] SEP-28k-E available: https://tinyurl.com/yck9fmfv

Table 1
Distribution of annotations of 3 second segments in the Kassel State of Fluency (KSoF) dataset where at least two annotators applied a given label (Bayerl et al., 2022).

Stuttering labels	KSoF	Description
Block	20.74%	Gasps for air or stuttered pauses
Prolongation	12.02%	Elongated syllable or Sound e.g., "[IIII]I", otherwi[ssss]se
Sound Repetition	14.76%	Repeated syllables "[nat-nat-nat-]naturally" or sounds "I [t-t-t-]talked to dad.
Word/Phrase Repetition	3.88%	"I have [I have] done no such thing"
No dysfluencies	24.75%	There are no audible dysfluencies
Modified/Speech technique	24.44%	Soft voice onset, at the start of syllables, voluntary prolongation with continuous phonation, e.g., rrReading, prrooolongation
Interjection	12.97%	Filler words, e.g., "ᵊahm", "ᶠah", "naja", eng: "uhm", "uh"
Non stuttering labels		
Natural pause	1.97%	A non-stuttered, significant pause in speech
Unintelligible	2.00%	The speech is difficult to understand
Unsure	0.30%	An annotator was unsure of their response
No Speech	0.39%	The clip contains no speech or is silent
Poor Audio Quality	0.98%	There are microphone or other quality issues
Music (Background Noise)	0.13%	Audible noise or music playing in the background

Table 2
Distribution of annotations of 3 second segments in the Kassel State of Fluency challenge (KSF-C) dataset (Schuller et al., 2022).

KSF-C: Label distribution (% (#))				
class	train	dev	test	Σ
Block	12.45 (310)	10.59 (104)	15.58 (176)	12.82 (590)
Interjections (Fillers)	8.24 (205)	10.39 (102)	7.35 (83)	8.48 (390)
Garbage	2.09 (52)	3.36 (33)	1.42 (16)	2.20 (101)
Modified/Speech technique	27.60 (687)	18.84 (185)	29.29 (331)	26.15 (1203)
Prolongation	7.35 (183)	5.40 (53)	9.73 (110)	7.52 (346)
Sound Repetition	6.79 (169)	3.87 (38)	11.68 (132)	7.37 (339)
Word/Phrase Repetition	2.13 (53)	2.34 (23)	1.59 (18)	2.04 (94)
No dysfluencies	33.35 (830)	45.21 (444)	23.36(264)	33.43(1538)
Σ	54.10 (2489)	21.34 (982)	24.56 (1130)	100.00 (4601)

3. Methods for stuttering classification

This section briefly gives an overview of the methods used by the challenge participants and in the experiments introduced in this paper. Fig. 1 contains a summary of methods and systems used by challenge participants.

3.1. LSTM

Long short-term memory (LSTM) neural networks are a type of recurrent neural network (RNN) that are able to learn long-term dependencies. Unlike feed-forward neural networks, RNNs have feedback connections. LSTMs were designed as an improvement over recurrent neural networks that allow the training networks to use longer sequences (Hochreiter and Schmidhuber, 1997). The LSTM gating structure helps to deal with the vanishing gradient problem. To achieve this, LSTM cells are composed of an input gate, an output gate, and a forget gate that manages which information contributes to the cell state (Hochreiter and Schmidhuber, 1997). LSTMs are particularly good in capturing long-term relationships within sequences, making them a logical choice for stuttering detection and classification, and have already been used for that purpose (Kourkounakis et al., 2020; Lea et al., 2021; Bayerl et al., 2022).

Bidirectional LSTM (Bi-LSTM) networks consist of at least two LSTM cells. The cells process the sequence in parallel. One of the cells processes the sequence in the designated order, while the other does so in reverse order. The outputs of both cells are concatenated, yielding a representation with two distinct views of the same data that are processed together.

3.2. Residual neural networks

Residual neural networks (ResNet) are a type of convolutional neural network (CNN) and have been proposed for image recognition (He et al., 2016). Residual connections as proposed by He et al. (2016) allow the training of very deep neural networks

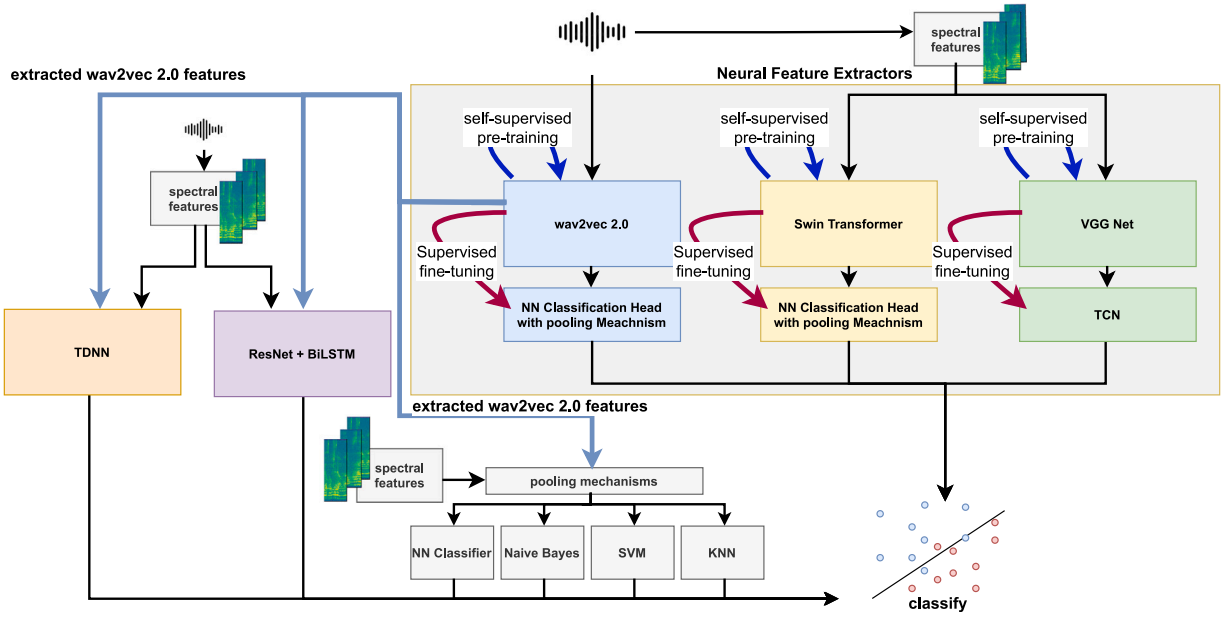


Fig. 1. Schematic overview of methods used in the 2022 ComParE challenge, subchallenge stuttering.

and are an essential building block of many neural network architectures. CNNs in general, are used to extract features by applying learned convolutional filters to an input signal. Unlike convolutional filters using a specific filter function, CNNs learn these filters by using gradient descent optimization. CNNs are versatile feature extraction systems that have applications in various image, signal, and speech processing tasks, including stuttering detection (He et al., 2016; Lim et al., 2016; Pepino et al., 2021; Bayerl et al., 2021; Wenninger et al., 2019; Kourkounakis et al., 2020).

CNNs in speech processing are mostly used in conjunction with spectral features that share image-like characteristics and can be visually inspected and interpreted by humans. These characteristics have been exploited by using CNNs pre-trained on image recognition tasks and then used for feature extraction from audio or fine-tuning models for audio tasks. Some systems operate directly on the raw waveform audio and extract features by applying stacked convolutional layers (Baevski et al., 2020).

3.3. VGGNet

Like ResNets, VGGnets are convolutional neural networks initially designed for image recognition tasks (Simonyan and Zisserman, 2015). They can be adapted for use with spectral features with minimal modifications, as shown by Hershey et al. (2017). Their approach aimed at creating a general-purpose audio feature extraction system by learning filters on large amounts of data. The VGGNet-based system (VGGish going forward) uses log mel spectrograms as inputs and was pre-trained on a large audio event dataset taken from YouTube (Gemmeke et al., 2017).

3.4. Time delay neural network

Time delay neural networks (TDNN) are designed to capture temporal relationships within a speech signal. The architecture is modeled in a way that the initial layers capture smaller contextual relationships and later layers capture a broader context (Peddinti et al., 2015). This is different from other neural network architectures that typically capture a broader context early in the processing hierarchy and smaller contextual relationships in later layers. The TDNN model uses statistical pooling before the classification layer.

TDNNs have been used as acoustic models in hybrid ASR systems and gained great popularity with the x-vector system in speaker recognition tasks. These TDNNs were modeled to capture speaker characteristics irrespective of the input sequence length (Chung et al., 2018; Snyder et al., 2018).

3.5. wav2vec 2.0

Wav2vec 2.0 (W2V2) is a neural network model for audio feature extraction based on the transformer architecture (Baevski et al., 2021; Vaswani et al., 2017). W2V2 is designed to learn general-purpose audio representations from large amounts of unlabeled training data. The model is usually pre-trained using a self-supervised training paradigm. Transformer models use self-attention blocks to make the model focus on the "most important" parts of a given input w.r.t. the training objective (Vaswani et al., 2017; Baevski et al., 2020).

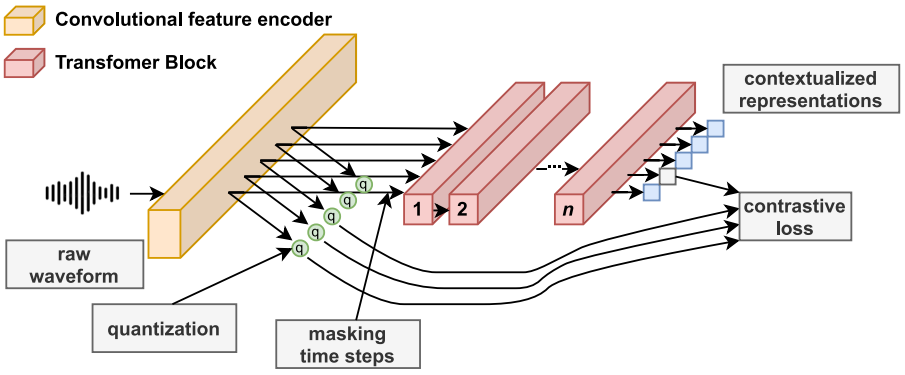


Fig. 2. Schematic overview of the wav2vec 2.0 pre-training process based on Baevski et al. (2020). The model takes a raw waveform input, first processed by a convolutional feature encoder that yields latent representations. These latent representations are then quantized and later used for the calculation of the loss. The n-transformer encoder layers process the latent features. A portion of the time steps is masked during training, while the network's objective is to predict the features at the masked time steps. The last transformer block yields contextual representations that are used to calculate a contrastive loss with the quantized representations. After pre-training, the model can be fine-tuned or used as a feature extractor taking in the raw waveform and yielding hidden representations after each transformer block.

Unlike many other speech processing systems, wav2vec 2.0 does not need spectral features as inputs as it operates directly on the audio signal (Gong et al., 2021). At the beginning of the model, there is a convolutional neural network (CNN) feature encoder that is trained in conjunction with the rest of the model. The feature encoder yields latent speech representations that are quantized during pre-training. The main wav2vec 2.0 training objective during pre-training is to predict the correct quantized representation q_t from the $K+1$ quantized candidate representations $\tilde{q} \in Q_t$ given some context $\mathbf{C}$, as is visualized in Fig. 2. The set Q_t consists of targets q_t and K distractors that are sampled uniformly from other masked time steps.

The latent representations are processed by n-transformer encoder blocks. At the last transformer encoder layer, the model yields contextualized speech representations c_t, which are used to compute the contrastive loss w.r.t. the quantized representations as shown in Eq. (1)

$$-\log \frac{\exp(\mathrm{sim}(c_t, q_t)/\kappa)}{\sum_{\tilde{q} \in Q_t} \exp(\mathrm{sim}(c_t, \tilde{q}_t)/\kappa)} \tag{1}$$

κ stands for the training temperature, which remains constant during training. The similarity (sim) between contextualized representations c_t and quantized representations q_t is calculated using the cosine similarity (Baevski et al., 2020). The training objective is extended by a code-book diversity loss which is a regularization technique that prevents the model from always favoring a trivial solution using only a few codes (Baevski et al., 2020).

There are multiple versions of wav2vec 2.0 models that are pre-trained and fine-tuned on data from different domains and languages. The most notable general distinction between models can be made upon the number of transformer encoder layers n and the dimensionality of the hidden representations. The most common models are the $W2V2_{Base}$ model consisting of 12 transformer encoder blocks with hidden representations $\mathbb{R}^{768}$, and $W2V2_{Large}$ consisting 24 transformer blocks and hidden representation of size $\mathbb{R}^{1024}$.

3.6. Swin Transformer

Swin Transformers (ST) are hierarchical vision transformer models (Liu et al., 2021). The ST limits self-attention to local windows instead of using global attention. The windowing approach helps to reduce computational cost and make the model scale linearly w.r.t. to the image size. In the ST approach, windows that limit self-attention are shifted depending on the location in the processing hierarchy. The shifting of those self-attention windows leads to hierarchical computation of self-attention, yielding distinctively different features depending on the location in the processing hierarchy. This approach allows computational efficiency while at the same time allowing for cross-window connections (Liu et al., 2021). While aimed primarily at image-processing tasks, the architecture can be adapted to other modalities.

3.7. Multi-task learning

Overfitting and a lack of generalization are problems when training stuttering classification systems, stemming from the relatively small datasets that only have little speaker variance (Sheikh et al., 2022b). Systems trained on data from one domain might perform poorly in another domain and vice versa.

Multi-task learning (MTL) is a regularization technique that can help avoid overfitting. As indicated by the name, systems trained using MTL learn more than one task, so-called auxiliary tasks. These tasks are related but different from the main task, which forces the model to generalize. The model can profit from cross-task regularization and auxiliary information. MTL has been used in various

speech tasks, such as speech recognition (Ravanelli et al., 2020), speech emotion recognition (SER) (Cai et al., 2021), and stuttering detection (Bayerl et al., 2022b; Lea et al., 2021; Sheikh et al., 2022a)

MTL can be implemented by modifying the loss function to combine two or more loss functions. In the simplest case, the loss can be formulated by adding the auxiliary loss L_{aux} to the main loss L_{main}, as shown in Eq. (2). The weights λ_{main} and λ_{aux} can be used to balance the contributions of each task to the final combined loss, with $1 > \lambda_{\text{main}} > 0$ and $\lambda_{\text{aux}} = 1 - \lambda_{\text{main}}$.

$$\mathbf{L}_{\text{MTL}} = \lambda_{\text{main}} L_{\text{main}} + \lambda_{\text{aux}} L_{\text{aux}} \tag{2}$$

3.8. Statistical classifiers

The KSF-C dataset is relatively small, with its only 4600 clips split into train, development, and test set, calling for algorithms that work well with only a few samples.

Support Vector Machines. Support vector machines (SVM) are a type of supervised learning algorithm that can be used for classification or regression (Boser et al., 1992). The goal of an SVM is to find the best boundary or decision surface that separates the data into classes given some constraints. This boundary is called a "hyperplane". The points closest to the hyperplane are called "support vectors". Once the hyperplane is determined, new data can be easily classified by seeing on which side of the hyperplane it falls. In the multi-class case, the problem is split into multiple binary classification problems, using a binary classifier per each pair of classes. SVMs are powerful because they can handle high-dimensional data and can be used with different kernel functions to find complex non-linear decision boundaries (Bishop, 2006).

Naive Bayes classifier. The Naive Bayes classifier (NBC) is a probabilistic machine learning algorithm that uses Bayes' theorem to predict the likelihood of a given outcome based on prior knowledge. In the context of the Naive Bayes classifier, the prior probability is the likelihood of a given outcome occurring based on the observed training data, and the likelihood is the probability of a given outcome occurring based on the features of the data. The Naive Bayes classifier assumes that all features are independent of each other, meaning that the probability of one feature occurring does not affect the probability of another feature occurring (Bishop, 2006). It is therefore called "naive" because this assumption often is not true in real-world data sets. The algorithm uses Bayes' theorem to calculate the probability that a given data point belongs to a certain class based on the probabilities of the individual features in the data set. The NBC uses the training data to estimate the probabilities of each feature occurring in each class, which are in turn used to predict the likelihood of the class of an unseen sample. The NBC is a simple and efficient algorithm that can be used in different applications due to its fast training and prediction times and its ability to handle large amounts of data.

K-Nearest Neighbors. K-Nearest Neighbors (KNN) is a non-parametric classification algorithm that uses the notion of distance between data points to classify new data points. The algorithm uses the training data to classify new data points. A sample is classified as the majority class of its K nearest neighbors, with K determining the number of neighbors used to classify a data point (Bishop, 2006).

4. Related work

4.1. Machine learning for stuttering detection and classification

Stuttering classification and detection have a long history in pattern recognition. There are various factors that limit the comparability of stuttering classification methods. Classification approaches differ w.r.t. the number of classes, the dataset used, the train, test, and development split of the datasets, the temporal context of the labels, and the modalities used. Early work by Howell et al. did use artificial neural networks to classify prolongations and word repetitions using spectral and autocorrelation function bases features in their acoustic detection approach (Howell and Sackin, 1995). Nöth et al. used ASR systems with modified pronunciation graphs to detect dysfluency events (Noeth et al., 2000). The authors of Lustyk et al. (2015) found evidence for the transferability of features in stuttering detection between languages by analyzing German stuttering recordings using algorithms designed and evaluated on Czech stuttering data (Lustyk et al., 2015). Esmaili et al. (2017) differentiate between fluent and prolonged speech in recordings of read speech taken from the UCLASS dataset (Howell et al., 2009) and a proprietary Persian dataset using SVMs with cross-correlation similarity measure and Perceptual Linear Prediction (PLP) features.

Kourkunakis et al. used LSTM networks with ResNets as a feature extraction backend for the detection of multiple types of stuttering (Kourkounakis et al., 2020). The authors of Bayerl et al. (2020) could show that time-aligned ASR output can be used to differentiate between fluent and non-fluent speech. FluentNet combines Squeeze and Excitation ResNets as a feature extraction backend with Bidirectional LSTM networks and an attention mechanism for stuttering detection (Kourkounakis et al., 2021). Lea et al. introduced the SEP-28k dataset and published baseline systems using LSTM networks with spectral-, F_0-based-, articulatory-, and phonemic features (Lea et al., 2021). Sheikh et al. introduced StutterNet, a TDNN-based system for stuttering classification (Sheikh et al., 2021). Bayerl et al. released the Kassel State of Fluency dataset, including baseline experiments using wav2vec 2.0 embeddings and LSTM networks (Bayerl et al., 2022). The authors of Bayerl et al. (2022c) extended the SEP-28k dataset with automatically generated speaker labels and showed the influence of data splitting on the performance of dysfluency detection systems.

The authors of Bayerl et al. (2022b) could show that large self-supervised pre-trained models yield features suitable for stuttering detection. Fine-tuning these models for English stuttering detection improves the performance of dysfluency detection systems not

only for English stuttering data but also improves stuttering detection in another language. In an attempt to improve the robustness of their stuttering detection systems, Sheikh et al. used adversarial learning and achieved promising results (Sheikh et al., 2022a).

Recent work w.r.t. stuttering was also considered with frame-level correction of artificially generated stuttered speech (Harvill et al., 2022), and the improvement of ASR systems for stuttered speech (Shonibare et al., 2022) The work by Zhang et al. focuses on creating artificial stuttering data to train better stuttering detection, and speech recognition systems tailored for PWS with only little performance degradation on non-dysfluent speech (Zhang et al., 2022).

4.2. The Computational Paralinguistics Challenge 2022: Stuttering

The Computational Paralinguistics Challenge (ComParE) is a series of open challenges in the field of Computational Paralinguistics that has taken place annually since 2009. It deals with states and traits of individuals that affect their speech signal properties. The challenge adds new tasks every year, showing that many highly relevant paralinguistic phenomena have not yet been covered. At the same time, the challenge introduces new baseline methods and challenge types and introduces new topics to the speech-processing research community.

4.3. Challenge contributions

An overview of all challenge contributions and results reported in accepted papers from the challenge can be taken from Table A.6. The deciding criterion used for the challenge is the unweighted average recall (UAR). This is done, as in a generic classification approach, performance should not be optimized towards specific classes, but rather towards the overall classification performance. UAR is the preferred measure in case of an unequal distribution of classes, as it helps to prevent a bias towards classes that appear frequently in the training data. The best results from each submission, ranked by test-set performance (unweighted average recall (UAR)), can be found in Table 3.

Table 3
Summary of best results of all contributions, including the baseline ranked by the test set unweighted average recall (UAR).

	Contributor	Results	
(#)		Dev	Test
1.	Grósz et al. (2022)	61.3	62.1
2.	Montacié et al. (2022)	–	52.3
3.	You et al. (2022)	42.0	46.4
4.	Sheikh et al. (2022c)	41.0	42.7
5.	Schuller et al. (2022)	28.1	40.4

The contribution by Sheikh et al. considered three distinctively different approaches; a vector-classification-based approach, an LSTM-based approach, and a TDNN-based approach (Sheikh et al., 2022c). They used either Mel Frequency Cepstral Coefficients (MFCC), features extracted from a frozen W2V2 base model that yields embeddings with $t \times \mathbb{R}^{768}$, or a combination of the later. The features from the W2V2 model were extracted directly after the convolutional feature extractor, which can be seen in Fig. 2, and after each of the 12 layers of the model. Before passing the extracted embeddings to the Support Vector Machine (SVM), Naive Bayes, K-nearest neighbor, and multi-branch shallow neural network (shallow NNet) classifiers, they apply statistical pooling along the time dimension. The resulting standard deviation and mean vector are concatenated and passed to the classifiers, resulting in a $\mathbb{R}^{1536}$ vector.

Shallow NNet is an ensemble of two three-layer neural network classifiers that share a common input and are joined in the final decision. This multi-branch (MB) approach is fashioned after the one described in Sheikh et al. (2021), where one branch predicts the decision fluent or non-fluent and the second branch differentiates between dysfluency types, once the fluent branch predicts the sample to be dysfluent.

Sheikh et al. also use an MB version of the ResNet+BiLSTM system based on the method introduced by Kourkounakis et al. (2020). The model uses a ResNet as a feature extractor and a bidirectional LSTM network to model the temporal information. All their neural networks use two modified weighted cross-entropy loss terms to deal with the class-imbalance of the dataset (comp. Table 2) that are combined to a multi-task loss (Sheikh et al., 2022c). The MB StutterNet is based on the TDNN architecture described in Section 3.4 (Sheikh et al., 2021). Compared to their initial publication, the authors use an MTL-based approach as described in Section 3.7.

The MB StutterNet and the MB ResNet+BiLSTM system were trained with either MFCCs or W2V2 input features. Sheikh et al. also explore the performance of W2V2 features extracted from different layers or a combination of those layers by either concatenating or summing up the vectors. Their best-performing system used summed W2V2 as input features for their MB StutterNet classifier.

You et al. also participated in the vocalization sub-challenge in addition to participating in the stuttering sub-challenge (Schuller et al., 2022). They introduced a rather general-purpose pre-training method for audio-signal classification based on the Swin-Transformers. The models used in the challenge were pre-trained on audio even classification data (Liu et al., 2021; Gemmeke et al., 2017). Their approach uses log mel spectrograms as inputs to the unsupervised pre-training of their models. During unsupervised pre-training the training objective is to predict masked patches in the log mel spectrogram. The patch-masking strategy proved to be the most successful masking method. Patch masking is supposed to lead the model to learn structural–contextual features, which

perform better in audio classification tasks instead of time-masking, which helps learn time-contextual features (You et al., 2022). The paper mentions the masking ratio, which was 0.6 with a patch size of 32 but leaves out implementation details such as the number of mel bins used, the frame width, and the frame step, which influence the size of the underlying spectrogram, and the dimensionality of the final features.

The pre-trained model is subsequently fine-tuned in an unsupervised manner on the KSF-C dataset to yield features suited for stuttering classification. The frozen model is then used as a feature extractor for the audio classification tasks. Based on the original implementation by the authors of the ST paper (Liu et al., 2021), the actual classification is performed using a single classification layer with softmax after global average pooling.

Montacié et al. extracted W2V2 features from pre-trained models. In their experiments, they used base and large models that yield embeddings of size $\mathbb{R}^{768} \times t$ and $\mathbb{R}^{1024} \times t$ per layer. Their approach treated the extracted W2V2 features as low-level descriptors (LLD), not unlike what openSMILE does with actual LLDs (Eyben et al., 2010) and computed multiple functionals over each feature dimension. The functionals chosen aimed at a balanced of estimating spatial- and temporal variability. Doing so yields 252 distinct feature sets.

To select the most relevant features, Montacié et al. investigated the feature sets taken from different layers and found features taken from layers 5 and 6 of the base model to be the most discriminating ones (Montacié et al., 2022). The most discriminating features were chosen by training multiple SVMs with linear kernel functions and picking the ones with the best UAR on the development set. The full feature sets yield 16 128 features for the base model per clip, which uses functionals over hidden states taken from one layer that are concatenated into a single vector, i.e. 21 functionals over feature vectors of $\mathbb{R}^{768} \times t$, leading to one vector of $\mathbb{R}^{16128}$. The full feature set for the base model yields 193 536 per clip, which consists of functionals computed for all hidden states, so a concatenation of twelve $\mathbb{R}^{16128}$ vectors. The advanced feature sets yield 15 360 features for the 15 most relevant functionals for the large model and 9984 for the 13 most relevant functionals for the base model. After feature extraction, the final feature vector is used in classification experiments with linear kernel SVMs.

Grósz et al. did focus on fine-tuning multiple pre-trained wav2vec 2.0 models. Besides their approach of fine-tuning W2V2 models, they explored the use of a CNN-based classifier that uses features extracted from a VGGish network (see Section 3.3) that was pre-trained on large amounts of audio event classification data extracted from YouTube (Abu-El-Haija et al., 2016). They also attempted to fuse their systems in an ensemble approach, but this did not outperform their best fine-tuned W2V2 model.

Two findings of the experiments were that models with more trainable parameters, in general, could achieve higher absolute performance w.r.t. UAR and using models pre-trained on data from the same language as the target task perform better. They could improve their performance through error analysis and ASR-based label correction. The systems trained with the KSF-C data often confused the ‘garbage’ label with the ‘no dysfluencies’ class, which can also be observed by looking at the confusion matrices of the test set results in Fig. 5. They used the length of the ASR-generated transcript to reclassify some of the samples classified as ‘no disfluencies’ as garbage using a threshold value which slightly improved their final results.

All results of papers submitted to the challenge and accepted for presentations were consolidated in Table A.6, and the top results on the test set ranked by UAR can be found in Table 3.

5. Experiments

The following section describes experiments that combine findings from the challenge with insights from current studies on stuttering detection and cross-language training of stuttering detection systems to improve the state-of-the-art results of the challenge.

5.1. Preliminary experiments

All challenge participants did use pre-trained self-supervised transformer models in their experiments, leveraging the information about spoken language that is encoded in these models by pre-training them on large amounts of data. The challenge could show that using W2V2 features and models yields superior results to other methods, which is also supported by prior work (Bayerl et al., 2022,b). The preliminary experiments were therefore performed to determine which W2V2 model to use for subsequent fine-tuning, as W2V2 models mainly differ w.r.t. the data they were trained on, the dimensionality of the internal representations, and the number of transformer encoder blocks used. The main experiments aim at solving problems that arise in a low-resource setting by evaluating multiple dataset compositions and pre-training schemes using a mixture of German and English stuttering data. In these experiments, W2V2 models were only treated as feature extractors. The experimental setup was similar to the one described in Bayerl et al. (2022b), Sheikh et al. (2022c).

The preliminary experiments were performed on the challenge task using only the challenge data. For the experiments, we chose the models from a base, and a large model trained on German data DE-ASR$_{base}$ and DE-ASR$_{large}$, a base model fine-tuned for ASR on English data, EN-ASR$_{base}$, and a multi-lingual model pre-trained on data from multiple languages, XLSR$_{large}$. The W2V2 models utilized were only used as feature extractors and were not previously fine-tuned on stuttering data. The model weights are open-source and can be obtained from the sources in Appendix C. The input for the SVM classifiers with linear kernel was computed as the mean of all vectors per 3-sec clip; using statistical pooling did not positively affect classification results. The hyperparameters for SVM training were optimized using extensive grid search on five-fold cross-validation on the combined train and development set. The grid search included a principal component analysis (PCA) on the W2V2 embeddings prior to SVM training. To deal with

Table 4
Unweighted average recall (UAR) results of preliminary experiments using W2V2 systems as feature extractors for stuttering classification using Support Vector Machines. Results on the development set are reported as the mean (standard deviation) of 5-fold cross-validation on the combined training and development set, including synthetic samples generated by the SMOTE algorithm, which in addition to speaker overlapping splits, leads to over-optimistic results on the development set.

(#)	Features	Dev	Test
1.	EN-ASR$_{base}$ L5	89.2 (3.9)	41.0
2.	DE-ASR$_{base}$ L5	89.1 (4.2)	47.6
3.	XLSR$_{large}$ L14	88.5 (3.8)	45.8
4.	DE-ASR$_{large}$ L12	90.0 (3.3)	**49.9**

the class imbalance, we used Synthetic Minority Over-sampling Technique (SMOTE) (Chawla et al., 2002) with a sampling factor of 50%, i.e., minority samples are synthetically generated for all non-majority classes to have a total of 50% of the majority class.

The number of principal components was chosen from $N_{\text{pca}} \in \{128, 256, 512, 768\}$ for features extracted from base models and $N_{\text{pca}} \in \{256, 512, 768, 1024\}$ for large models. The kernel parameter γ was selected from the set $\gamma \in \{10^{-k} \mid k = 3, 2, 1\} \subset \mathbb{R}_{>0}$, and The penalty parameter of the error term C was selected from $C \in \{1, 10, 100\} \subset \mathbb{N}_{>0}$, and the extraction layer L was selected from $L \in \{1 \dots 12\} \subset \mathbb{N}_{>0}$ for the base models, and $L \in \{1 \dots 24\} \subset \mathbb{N}_{>0}$ for the large models. Results for these experiments can be found in Table 4.

From the results of the preliminary experiments, two conclusions can be drawn. The first conclusion is that the language of the data used to train the model and its size matter. Using the larger $\mathbb{R}^{1024}$ embeddings extracted from the large models leads to better results than using the $\mathbb{R}^{768}$ embeddings extracted from the base models, supporting the observations by Grósz et al. (2022). Result #1 in Table 4 matches the results reported by Sheikh et al. using the German base model outperforms this system by 6.6% points UAR. Using advanced pooling strategies like Montacié et al. yielded performance improvements comparable to choosing a W2V2 model in the same language as the challenge data (see Table 3, Table 4). As a direct consequence of the preliminary experiments, we used the weights of the feature extractor that led to the best classification results, DE-ASR$_{large}$, as the basis for all subsequent end-to-end experiments.

The column 'dev' shows the mean and standard deviation over the results of the five cross-validation splits that are vastly optimistic compared to the results reported on the development set (comp. Table A.6) and the results of the final model on the test set (comp. Table 4). This has two reasons, due to the mixing of train and development set, the speaker exclusivity cannot be held up using 5-fold cross-validation on the combined train and development during grid-search, and the use of SMOTE also introduces bias to the data that skews results on the development set.

5.2. Main experiments

Labeled data for stuttering detection and classification, in general, is scarce. Labeled stuttering data for a specific language, such as German, are even scarcer (Sheikh et al., 2022b). Therefore, using pre-trained models such as W2V2 and similar training data, if available, are logical steps in solving acoustic stuttering classification.

The authors of Bayerl et al. (2022b) could show that W2V2 models fine-tuned on English stuttering data yield features that are suitable for stuttering detection in German stuttering-therapy recordings. Using the fine-tuned features in detection experiments also showed a language component in stuttering classification, especially w.r.t. word repetitions, which is supported by the results of the preliminary experiments. The regularizing effect of multi-task learning on stuttering detection could also be shown previously and in challenge contributions (Lea et al., 2021; Sheikh et al., 2022c; Bayerl et al., 2022a).

As described in Section 2, there are two other datasets containing labeled stuttering data with stuttering labels compatible with the KSF-C dataset. To match the training conditions as closely as possible when pre-training, we created additional dataset splits for pre-training and fine-tuning of W2V2 models using the FluencyBank and SEP-28k-E corpora. Just like with KSF-C, we removed all segments labeled with more than one type of stuttering. Filtering these clips from FluencyBank and SEP-28k-E leaves ∼2800 and 19300 clips, respectively. Segments labeled with either 'Unintelligible', 'No Speech', or 'Poor Audio Quality' were labeled as belonging to the 'garbage' class. The data was only used for pre-training and fine-tuning of models, not for testing; therefore, we used all the data belonging to the training and development sets for model training and the test partitions for validation (Bayerl et al., 2022c). The label distribution of the filtered additional datasets can be found in Appendix D.

Our models were trained using two approaches. The first approach employed a classical pre-training and fine-tuning method by pre-training the model using English stuttering data, of which there is a larger quantity, and a subsequent fine-tuning step using the German KSF-C data. As SEP-28k-E and FluencyBank do not contain labeled sections of modified speech, initial experiments only considered seven classes during pre-training. The other approach uses combinations of English and German training data with the aim of creating a more robust classifier and dealing with the small size of the dataset. As previous work has shown that features fine-tuned on English stuttering data can improve recognition results in German stuttering data for all classes, but word repetitions (Bayerl et al., 2022b).

The KSF-C dataset is imbalanced. The combined train and development portion of the dataset only contains 76 word repetitions, making it the rarest class in the KSF-C dataset. Initial experiments on the KSF-C data showed that models primarily struggled to recognize 'garbage' and 'word repetitions'. Experiments using W2V2 models on English training data could show that these models are capable of learning to detect word repetitions, given enough training data (Bayerl et al., 2022a). We can assume that with very diverse and complex patterns, more training data will improve robustness and generalization up until a certain point (Lei et al., 2019). We, therefore, used oversampling of word repetitions from the KSF-C data to match the number of the majority class in the KSF-C dataset, which is 'no dysfluencies'.

The dataset combinations used in our experiments were:

- KSF-C, the dataset split as provided by the challenge.
- KSF-C-Garbage adds all clips from the FluencyBank and SEP-28k-E labeled as 'garbage' to the KSF-C train and development partitions, making up for the low volume of 'garbage' labeled clips in the KSF-C dataset.
- EN-ALL combines all clips belonging to the train and development split of SEP-28k-E and FluencyBank[6] to a combined all English training set and uses the test partition as the development set. This split has seven classes.
- EN-ALL-M, uses an identical dataset split as EN-ALL but adds all samples labeled as 'modified' from KSF-C to the respective partitions. Extending the dataset by an eight class.
- FULL, merges the EN-ALL split with the KSF-C split by combining the respective partitions with each other.
- FULL-DE-DEV, consists of all English clips as the training data, including the KSF-C training data. The development and test partition are identical to the KSF-C partitioning, leading to a training set consisting of 24591 samples.

The splits EN-ALL and EN-ALL-M are solely used for pre-training of W2V2 models using the DE-ASR$_{large}$ model as the initial model weights, yielding W2V2 models trained for stuttering classification, EN-STUTTER$_{large}$ and EN-STUTTER-M$_{large}$. This is visualized on the left side of Fig. 3. The splits FULL and FULL-DE-DEV were used in experiments using joint training data. KSF-C and KSF-C Garbage were used for training and fine-tuning of models based on the previously trained models and DE-ASR$_{large}$. This process is visualized on the right side of Fig. 3.

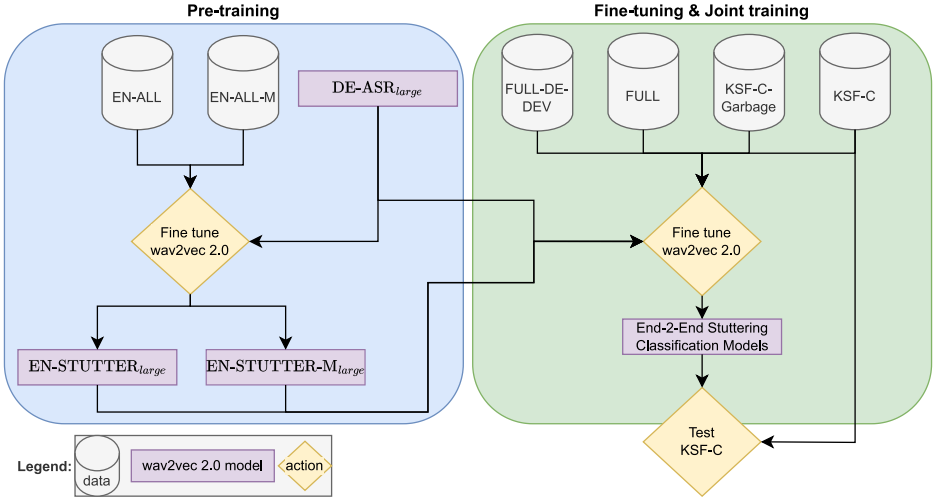


Fig. 3. Block diagram of pre-training, fine-tuning, and cross-dataset training of stuttering classification systems using multiple compositions of the available training data.

Unlike the full KSoF release, the challenge data comes with little meta-data and no speaker information per clip. This prohibits using speaker classification or gender classification as an auxiliary task, as used in Bayerl et al. (2022b). This leaves using the pseudo 'any' label as used by Lea et al. (2021) and Sheikh et al. (2022c), and 'language' when using a combination of German and English training data. One objective of an auxiliary task in MTL is to teach the model feature representations that are advantageous to the primary task. To achieve this, the task should be related to the main task; otherwise, the performance of the main task suffers (Caruana, 1998). Using 'language' classification as an auxiliary task is a logical choice, as the information is available, and previous experiments have shown that language plays a role in classifying stuttering (Bayerl et al., 2022b). While blocks, sound repetitions, and prolongations are relatively similar, independently of the language, the realization of word repetitions is less similar.

Pre-training and fine-tuning experiments were performed using the mean-pooling-over-time-based STL and MTL classification head described in Bayerl et al. (2022b). Joint training, pre-training, and fine-tuning experiments were performed using STL and MTL learning. Fig. 3 contains a block diagram showing which data and model were used at each step of the training of the end-to-end stuttering classification systems.

[6] https://gist.github.com/patientzero/b23943107d8c6f17cbcb5ffc13fd45b9

5.2.1. Pre-training on stuttering data

The DE-ASR$_{large}$ used as the basis for pre-training on stuttering data was already pre-trained on large amounts of unlabeled audio data and subsequently fine-tuned for ASR, so technically speaking, we are not pre-training, but rather fine-tuning the models for stuttering classification in two stages. During the first stage, referred to as pre-training here, we trained the model for stuttering classification using English training data. During pre-training, we considered two cases. The first case uses only data from the SEP-28k-E and FluencyBank datasets that were combined in the EN-ALL data split. Both datasets do not contain speech labeled as 'speech modifications' in them, making it a seven-class rather than an eight-class problem as the KSF-C datasets. Large Models pre-trained with them are denoted as EN-STUTTER$_{large}$. The other case uses the EN-ALL-M split containing all the English training data together with all clips marked as 'speech modifications' from the KSF-C dataset. The rationale behind adding the modified sample is to prepare the model for an eighth class, reusing all weights of the model, including the classification head. The models fine-tuned with EN-ALL-M are denoted as EN-STUTTER-M$_{large}$.

All models were pre-trained for up to 10 epochs using the AdamW optimizer and a batch size of 32. The warm-up phase consisted of 10% of the total training steps with an initial learning rate of 3×10^{-5} using a frozen convolutional feature extractor at the beginning of the model. Early stopping was implemented w.r.t. to the development loss with a patience of 4. The models pre-trained using MTL used the 'any' label as an auxiliary task with $\lambda_{\text{main}} = 0.9$.

5.2.2. Fine-tuning and Joint training

Fine-tuning is either the second stage to pre-training described in Section 5.2.1 or directly starts by fine-tuning the DE-ASR$_{large}$ model without fine-tuning on stuttering data first. Joint training describes the process of fine-tuning the same W2V2 model using the cross-dataset splits defined in Section 5.2.

All models were fine-tuned for up to 10 epochs using the AdamW optimizer, a batch size of 8, an initial learning rate of 3×10^{-5}, and a warm-up phase that consists of 10% of the total training steps. The convolutional feature extractor at the beginning of the model remained frozen during training. Early stopping was implemented w.r.t. to the unweighted average recall on the development set with a patience of 4.

MTL training was performed using 'language' recognition as the auxiliary task for joint training with the FULL and FULL-DE-DEV splits and with the 'any' label for all other cases. The best results were achieved using $\lambda_{\text{main}} = 0.6$ in the case of joint training and language recognition and $\lambda_{\text{main}} = 0.8$ in the case of fine-tuning using 'any' as the auxiliary task.

5.3. Results

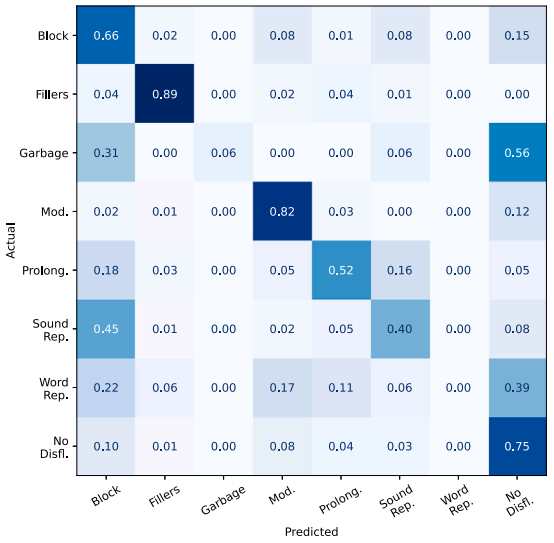
Results for the experiments described in Section 5.1 can be found in Table 4, and Table 5 contain results for the experiments described in Section 5.2. More detailed results for the experiments can be found in Appendix B.

Table 5
Unweighted average recall (UAR) results for fine-tuning various wav2vec 2.0 systems using STL and MTL training with different training data combinations and three different model weights. Results in the column 'Dev*' refer to the results on the composition of the development according to the column 'Train Data'.

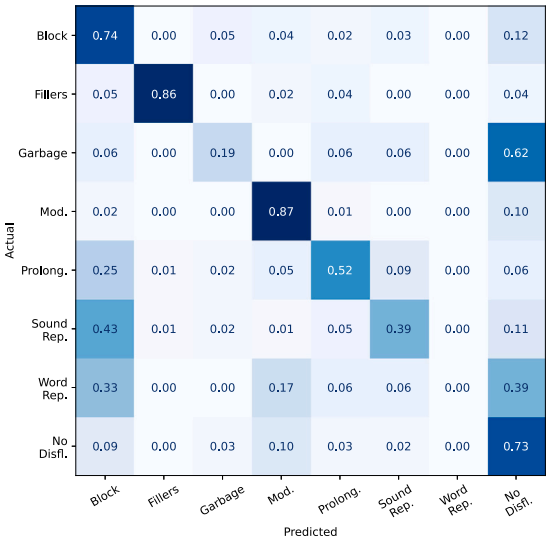
Main results					
(#)	wav2vec 2.0 model	Training scheme	Train data	Dev*	Test
1	DE-ASR$_{large}$	STL	KSF-C	47.6	51.2
2	DE-ASR$_{large}$	STL	KSF-C-Garbage	55.3	53.5
3	DE-ASR$_{large}$	STL	FULL	56.9	59.6
4	DE-ASR$_{large}$	STL	FULL-DE-DEV	54.0	60.9
5	EN-STUTTER$_{large}$	STL	KSF-C-Garbage	59.7	59.5
6	EN-STUTTER-M$_{large}$	STL	KSF-C-Garbage	61.2	**64.5**
7	like #6 + TBC			61.2	64.5
8	DE-ASR$_{large}$	MTL	KSF-C	49.1	58.3
9	DE-ASR$_{large}$	MTL	KSF-C-Garbage	56.4	59.2
10	DE-ASR$_{large}$	MTL	FULL	58.1	62.8
11	DE-ASR$_{large}$	MTL	FULL-DE-DEV	51.7	60.6
12	EN-STUTTER$_{large}$	MTL	KSF-C-Garbage	55.8	60.8
13	EN-STUTTER-M$_{large}$	MTL	KSF-C-Garbage	58.9	**67.2**
14	like #13 + TBC			58.9	67.2

All W2V2-based systems manage to beat the ComParE baseline by a substantial margin of at least 10 percentage points. The overall best results on the test set were achieved using MTL and a two-stage pre-training process of first fine-tuning the model on the EN-ALL-M data split and subsequent fine-tuning using the KSF-C-Garbage data split (Table 5 #13). When using the same pre-training strategy and training data, the MTL-trained models consistently outperform the models trained using STL.

Using cross-dataset training is a viable option for stuttering classification. The system trained using MTL, and the cross-language data even slightly outperformed the best challenge contribution by Gròsz et al. (comp. exp #10). Comparing the confusion matrix of the best cross-dataset trained model Fig. 4c with the confusion matrix of the overall best results in Fig. 4d reveals that the gains w.r.t. UAR mostly stems from better recognition of the 'garbage' and the better recognition of 'blocks'. In comparison to the results



(a) Confusion matrix for exp. #1. (UAR: 51.2%)



(b) Confusion matrix for exp. #2. (UAR: 53.5%)

(c) Confusion matrix for exp. #10. (UAR: 62.6%)

(d) Confusion matrix for exp. #13. (UAR: 67.2%)

Fig. 4. Confusion matrices for test set predictions using the models from experiments #1, #2, #10, and #13 from Table 5. Showing the effect of adding additional garbage samples to training data by comparing Fig. 4a and 4b and comparing results between using the FULL dataset split and the multi-stage fine-tuning approach by comparing Fig. 4c and 4d.

of the best overall performing model, it can be seen that the recall performance on sound repetitions is substantially better using cross-lingual training. This shows that sound repetitions profit from a wider variety of training data, as the pattern is very similar across languages.

The 'Filler' class is very distinguishable and interjections are similar between English and German when it comes to their purpose and phonetic composition (Belz, 2021). Recognition rates are high across all experiments and only slightly profit from changes in the training data or paradigm.

To our surprise, the performance of the cross-dataset trained model and the best-performing model did not affect performance on prolongations, which is also similar across languages. This differs from a previous study that considered the multi-label dysfluency detection task, where cross-language training also improved classification results on German prolongations (Bayerl et al., 2022a). Fine-tuning still leads to performance gains w.r.t. the detection performance supporting the results in Bayerl et al. (2022b), but recall results still do not improve over 65%.

Comparing the confusion matrices from experiment #1 and #13 from Table 5 in Fig. 4 shows improvements across all classes. Bigger models, multi-stage fine-tuning, and the modification of the training data all contribute to improved results. The hardest class to recognize for W2V2-based models is word repetitions. The pattern is language-dependent, as shown by Bayerl et al. (2022a), and needs a different temporal context than the other patterns to be recognized. Therefore recognition results probably suffer the most using mean pooling over time and only slightly improve when using the multi-stage fine-tuning method or cross-language training. In most cases, except for the best-performing model, the primary confusion class for word repetitions is no dysfluencies, which is to be expected, considering that simply repeating words is acoustically indistinguishable from fluent speech. This exception is quite peculiar, as, in experiment # 13, most word repetitions are confused with Modifications, Sound Repetitions, and the Garbage class.

The best-performing model w.r.t. UAR on the test set comes with the drawback of having a worse recall on the majority class 'no dysfluencies'. 'No dysfluencies' get confused the most with 'Modifications', which can be expected, as the aim of the fluency shaping technique is to sound as natural as possible, which is reflected in mutual confusion. 'Modifications' are similarly confused the most with the 'no dysfluencies' class when looking at Fig. 4d.

6. Discussion

This section places challenge contributions, discusses the methods used, relates them to the experiments in this paper, gives an overview of challenges, and an outlook to future work.

6.1. Discussion of methods and challenge results

A major trend that could be observed in other pathological and paralinguistic topics is the use of models that have been pre-trained using the SSL-paradigm. The challenge contributions are no exception to this trend as all did use pre-trained models. Transformer models have become state-of-the-art in many speech-processing areas. Particularly low-resource tasks, such as stuttering classification, have profited from the use of these models a lot. Three of four contributions used W2V2, which follows the trend of using W2V2 as a general-purpose feature extractor for all kinds of speech processing tasks. This might have been slightly influenced by the best-performing baseline for the stutter detection task of the initial KSoF release, which used W2V2 features (Bayerl et al., 2022).

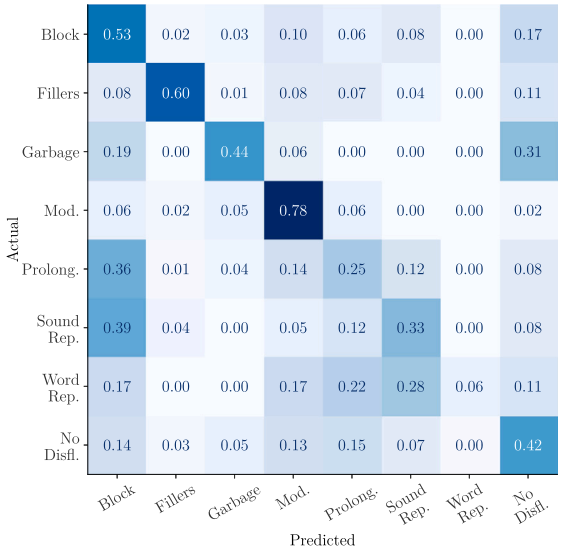
The approach taken by You et al. did use an SSL pre-trained model but still was substantially different from the other approaches. The ST-based system they used was not solely pre-trained on human speech data but rather on a dataset for audio-event detection (You et al., 2022; Gemmeke et al., 2017). Instead of operating directly on the raw audio waveform using a convolutional feature extractor, their approach employed log mel-spectrograms as inputs. The pre-training used did not mask quantized representations as W2V2 but masked out regions in the spectral features during pre-training. Their approach beat the baseline but fell short compared to the specialized speech-transformer fine-tuned in this study and by Grósz et al. (2022). Looking at the confusion matrix for the best-performing system by You et al. in Fig. 5b reveals that their approach had the lowest test-set recall on the garbage class. This is unexpected, as the audio-event data the system was pre-trained on has a lot of audio samples that are close to the contents of the garbage class.

In general, the system seems to have favored the majority class 'no dysfluencies', which can be seen by the majority of confusion across most classes towards the 'no dysfluencies' class. Ultimately the system worked better than the baseline and has shown potential for dysfluency classification. Pre-training with more speech data instead of audio-event data might have contributed to overall better results in the challenge, as well as taking cost-function-based measures to deal with the class imbalance.

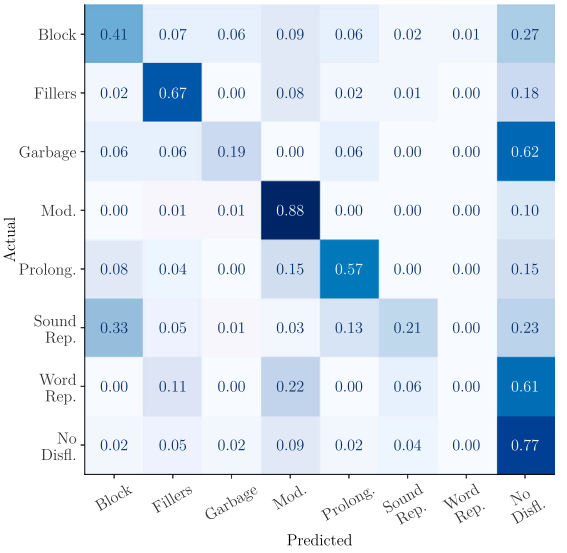
The contribution by Montacié et al. used different sizes and SSL models and mostly worked on different pooling strategies for the sequence data to be classified. In their approach, they treated each of the W2V2 systems as yielding audio LLDs and used combinations of statistical functionals in their pooling approaches instead of simply using mean- or statistical pooling. Their approach outperformed the baseline by a significant margin w.r.t. to the UAR on the test set. The actual effect of the method is unclear and should be explored using a more suitable model than models solely pre-trained on English data. Comparing the results from Table 4 shows that the best result using advanced pooling strategies only beats the best results in our preliminary experiments that only employ mean pooling by 2% points. The effect the pooling strategies have is about as large as the effect that choosing better-suited W2V2 weights has.

Fine-tuning of W2V2 models has, by a margin of 9.8% points UAR to the second-best performing method, the largest positive effect on the dysfluency classification results. Choosing a suitable pre-trained model for fine-tuning is crucial for good performance, as shown by Grósz et al. (2022). Another beneficial factor was the number of trainable parameters. Their W2V2 experiments were, in principle, simple and straightforward, which shows the power of the underlying method. The problem of stuttering classification is still not solved, as shown by the fact that a simple transcript- and threshold-based method to improve the final results, as employed by Grósz et al. (2022). Their final gains could be achieved by using a rather simple method based on the length of automatically generated transcripts which exploited the length to differentiate between the garbage class and 'no dysfluencies.'

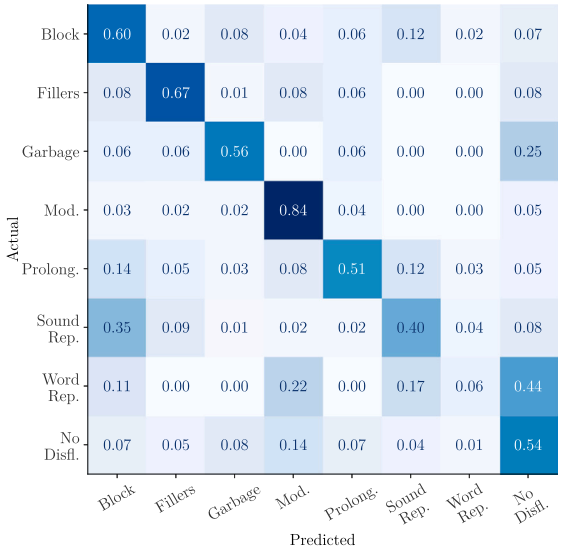
The additional experiments in this paper could show making changes to the training data and using MTL as a regularization technique unlocks some more potential of the wav2vec 2.0 method, even without using a model with 48 transformer layers such as the one used in Grósz et al. (2022). Applying text-based correction (TBC) as described in Grósz et al. (2022) to our best STL and MTL trained models does not improve these results (Table 5 #7, #14). The models already gained extra classification performance on the 'garbage' class through regularization and modifying the training data.



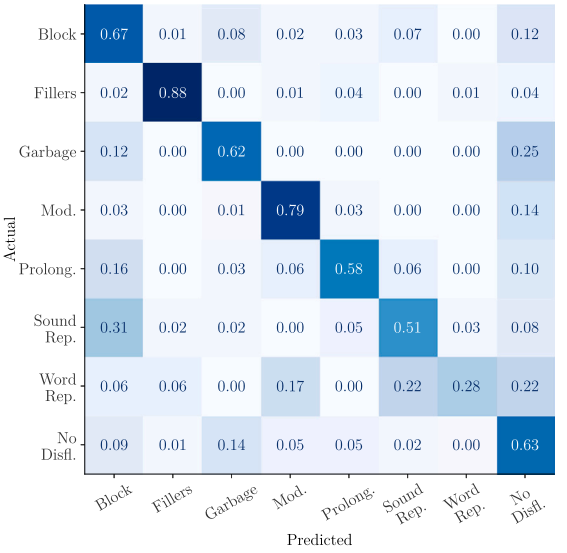
(a) Sheikh et al. (UAR: 42.7%) (Sheikh et al., 2022)



(b) You et al. (UAR: 46.4%) (You et al., 2022)



(c) Montacié et al. (UAR: 52.3%) (Montacié et al., 2022)



(d) Grósz et al. (UAR: 62.1%) (Grósz et al., 2022)

Fig. 5. Test-set confusion matrices for the best systems (w.r.t. UAR on the test set) for all challenge participants as summarized in Table 3.

The challenge contributions could classify 'Fillers' consistently well, and word repetitions were consistently the class with the lowest recall performance (comp. Fig. 5). As shown by Grósz et al. (2022), combining the different W2V2-based methods does not lead to improvement w.r.t. to those classes. Different methods will have to be employed to gain performance in recognizing word repetitions. Possible solutions to this problem could be phoneme recognition or ASR-based approaches. Especially the ASR-based approach would require ASR systems that can consistently transcribe word repetitions and incomplete words that are not removed due to a language model.

6.2. Task

Challenges are a great way to introduce problems that can be solved with machine learning to the broader research community. The time to work on a challenge is limited, and therefore, challenge organizers need to put some constraints on a task. In the case

of the 2022 ComParE stuttering sub-challenge, the KSoF dataset was adapted only to contain unambiguously labeled clips (Schuller et al., 2022), which represents a simplified view of stuttering. Even when only evaluated on short speech samples, like the 3-second long audio clips used in the sub-challenge, it is not uncommon to observe more than one type of core-stuttering behavior. Comparable datasets such as SEP-28k-E, FluencyBank, which are labeled similarly, and the full KSoF release contain a substantial number of clips that do not belong to a single class of stuttering (Bayerl et al., 2022a). They contain about 30%, 36%, and 21% ambiguously labeled clips.

Another limitation of the challenge task is how the best performance is measured using a single summary metric as the deciding criterion. A single metric can never deliver a full picture of the overall performance of a specific method on a specific task. The unweighted average recall used is a compromise that is easy to interpret and makes comparing the performance of models easy. It is not biased towards any particular class since it is calculated as an average of the recall values for each class without weighing any class over the others. This is useful when all classes are equally important. UAR is sensitive to changes in the performance of any class, which can be useful for identifying areas of improvement for a model. The strength of UAR is also its weakness, as it does not take into account the relative frequency or importance of the classes. If some classes are more important or more common than others, UAR may not accurately reflect the model's overall performance. UAR may not be sensitive enough to changes in the performance of classes with many samples and not accurately reflect the overall performance. A good, yet still simple, additional metric to take into account is the F-score, as it balances the importance of precision and recall.

A comprehensible view of the performance of a model for a specific classification problem can only be gained by looking at the confusion matrix. Trade-offs and acceptable failures can also only be interpreted and discussed w.r.t. a specific application and application context in mind. Nevertheless, the challenge task is an important step towards fully automated self-help therapy applications, being able to classify speech as fluent, belonging to one of the core behaviors, an off-class, or as a speech pattern acquired during therapy. The 2022 ComParE challenge raises awareness for the research problem of stuttering classification. Sharing the complete challenge data, including the test set labels, with the research community creates opportunities for new research, enables researchers to reproduce and improve upon the challenge contributions, and can help to facilitate innovation in the field.

6.3. Challenges and future work

The task of classifying and detecting stuttering behaviors from an acoustic signal is a difficult one. The number of patterns with different characteristics that occur within acoustic contexts of different lengths, the strong speaker- and situational dependence make the task challenging even for humans.

Identifying all behaviors correctly is difficult, as evidenced by inter-rater reliability metrics reported by dataset creators as low as Fleiss' κ of 0.11 for prolongations, or 0.25 for blocks on SEP-28k (Lea et al., 2021). The quality of the labels limits how well the systems can learn, as ultimately, this comes down to the quality of the training data. Therefore, creating a larger quantity of training data is insufficient. Creating quality stuttering data and validating the quality of existing data is crucial to advance research further. One way to increase the amount of training data might be the creation of language-specific, artificially generated stuttering data as suggested by Kourkounakis (2021). Assessing and increasing the quality could be done by applying an active learning strategy to identify low-quality labels and relabel them to increase the quality of the training data (Settles, 2012).

Classification results differ strongly depending on the dysfluency type. The confusion matrix in Fig. 4d shows recall on the 'Filler' class of 90%, whereas the performance on word repetitions is poor with a recall of just 33%. In the case of the challenge, this is surely also an issue with the number of available training samples, but framing this as the only issue would not be correct considering the results on the detection of word repetition with an F1-score of only 0.56 in Bayerl et al. (2022a). The large differences in the patterns to be detected with one and the same system also plays a role. A possible solution to account for the large variability of stutter phenomena could be achieved by combining multiple feature types and expert systems that handle specific classes. This could be done text-based for word repetitions but comes with the precondition of needing an ASR system that can reliably transcribe the speech of PWSs, including word- and sound repetitions. Future research should therefore be considered with creating such ASR systems.

Another possible way to account for differences in patterns could be to skip the word-level decoding and use phoneme recognition systems (PRS) such as the one described in Klumpp et al. (2022). The outputs of PRS contain information about structural- and content-based differences between normal and dysfluent speech. Combining these different systems and features for stuttering detection will be the subject of future work.

To make stuttering classification and detection models more useful in a clinical setting, systems need to be able to handle multiple speakers at once, continuous speech, and detect and classify multiple types of stuttering in a short time span.

7. Conclusion

This paper provides a comprehensive overview of stuttering classification in the context of the ACM ComParE challenge 2022. It shows the continuing trend of using large pre-trained neural feature extractors in low-resource scenarios. The challenge contributions and the experiments in this paper could also show the importance of the choice of the model weights, w.r.t. the data SSL-models were pre-trained on. In our experiments, we improved upon the challenge contributions by combining insights gained from prior work and challenge contributions. Applying more advanced learning techniques, such as multi-task learning, boosts performance

when fine-tuning SSL-based models for stuttering classification. Furthermore, we could show potential for cross-dataset and cross-lingual training for stuttering classification. By publishing the full challenge data for the research community, we made researching stuttering classification more accessible to new researchers who can now reproduce and improve upon our work and the challenge contributions.

CRediT authorship contribution statement

Sebastian P. Bayerl: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. **Maurice Gerczuk:** Validation, Writing – review & editing. **Anton Batliner:** Writing – review & editing. **Christian Bergler:** Writing – review & editing. **Shahin Amiriparian:** Writing – review & editing. **Björn Schuller:** Writing – review & editing. **Elmar Nöth:** Supervision. **Korbinian Riedhammer:** Supervision, Funding acquisition, Resources, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix A. Consolidated challenge results

Table A.6
Reporting ACM ComParE challenge 2022 baseline and contribution results as unweighted average recall (UAR) for the eight-class stuttering classification problem on the development (Dev) and test partition of KSF-C.

Method	Partition	
	Dev	Test
Baseline		
DeepSpectrum (Schuller et al., 2022)	28.1	40.4
BoAWs (Schuller et al., 2022)	26.7	32.1
Challenge contributions		
SVM (wav2vec 2.0, Layer 5) (Sheikh et al., 2022c)	36.9	41.0
SVM (wav2vec 2.0, Layer 5 + MFCCs) (Sheikh et al., 2022c)	37.6	42.6
MB StutterNet (wav2vec 2.0, Layer 5) (Sheikh et al., 2022c)	38.5	40.3
MB StutterNet (wav2vec 2.0, Layer 5:9) (Sheikh et al., 2022c)	40.8	42.6
MB StutterNet (wav2vec 2.0, $\Sigma_i L_i$) (Sheikh et al., 2022c)	41.0	42.7
Scratch (You et al., 2022)	29.0	–
SSAST (You et al., 2022)	38.7	–
Masked Swin-T (You et al., 2022)	42.0	46.4
W2V2 basic (S6) (Montacié et al., 2022)	–	48.1
W2V2 basic (Montacié et al., 2022)	–	48.2
W2V2 advanced (Montacié et al., 2022)	–	49.7
W2V2 large advanced (Montacié et al., 2022)	–	52.3
TCN (Grósz et al., 2022)	26.4	–
$wav2vec2_S^{de}$ (Grósz et al., 2022)	51.2	–
$wav2vec2_L^{de}$ (Grósz et al., 2022)	54.5	–
$wav2vec2_L^{de}$ frozen Transf. Grósz et al. (2022)	14.3	–
$wav2vec2_M^{de}$ (Grósz et al., 2022)	50.1	57.1
$wav2vec2_L^{de}$ (Grósz et al., 2022)	59.3	61.5
$wav2vec2_L^{de}$ (Grósz et al., 2022)	**61.3**	**62.1**
$wav2vec2_L^{de}$ full train (Grósz et al., 2022) + TCN	58.7	–
$wav2vec2_L^{de}$ full train + interpretability TCN + text-based correction (Grósz et al., 2022)	61.1	61.9
ensemble of all 3 (Grósz et al., 2022)	59.7	–
ensemble of all 3 + text-based correction (Grósz et al., 2022)	60.7	–

Appendix B. Detailed classification results

Table B.7
Table reporting detailed recall results for all experiments described in Section 5.

#	UAR	Block	Fillers	Garbage	Mod.	Prolong.	Sound Rep.	Word Rep.	No disfl.
1	51.22	65.91	89.16	6.25	81.87	51.82	40.15	0.00	74.62
2	53.51	73.86	85.54	18.75	86.71	51.82	38.64	0.00	72.73
3	59.57	43.18	77.11	18.75	**91.24**	60.91	81.82	27.78	**75.76**
4	60.93	63.64	77.11	25.00	78.85	**69.09**	64.39	**38.89**	70.45
5	59.49	69.32	74.70	37.50	87.31	45.45	73.48	22.22	65.91
6/7	64.51	**77.27**	**91.57**	43.75	79.76	64.55	59.85	27.78	71.59
8	58.26	69.32	**91.57**	25.00	89.73	60.00	50.00	11.11	69.32
9	59.22	70.45	89.16	56.25	89.43	60.91	47.73	0.00	59.85
10	62.78	47.73	86.75	31.25	**91.24**	65.45	**83.33**	22.22	74.24
11	60.61	65.34	83.13	25.00	87.61	61.82	73.48	22.22	66.29
12	60.84	68.18	85.54	31.25	80.36	61.82	74.24	11.11	74.24
13/14	**67.20**	57.95	90.36	**68.75**	89.73	64.55	72.73	33.33	60.23

Table B.8
Table reporting detailed precision results for all experiments described in Section 5.

#	UAP	Block	Fillers	Garbage	Mod.	Prolong.	Sound Rep.	Word Rep.	No disfl.
1	62.79	48.54	84.09	**100.00**	85.22	64.04	54.08	0.00	66.33
2	56.48	50.58	**97.26**	13.64	86.19	69.51	68.00	0.00	66.67
3	63.06	60.32	84.21	14.29	87.79	70.53	60.34	55.56	71.43
4	65.56	51.61	88.89	33.33	89.08	61.29	70.25	63.64	66.43
5	64.75	56.48	92.54	17.14	85.76	**75.76**	62.99	57.14	70.16
6/7	**69.66**	55.51	91.57	23.33	**92.96**	64.55	**75.24**	**83.33**	70.79
8	64.21	57.55	91.57	36.36	86.34	56.90	63.46	50.00	71.48
9	56.23	52.54	89.16	34.62	86.80	60.36	50.40	0.00	75.96
10	67.45	**65.12**	94.74	23.81	87.54	67.29	57.29	66.67	77.17
11	69.06	56.10	92.00	80.00	86.05	61.82	69.29	36.36	70.85
12	67.87	60.91	95.95	29.41	90.17	63.55	70.50	66.67	65.77
13/14	64.70	63.75	83.33	20.37	84.38	62.28	64.00	60.00	**79.50**

Appendix C. Open source models

Table C.9
Sources for open source wav2vec 2.0 models used in the experiments. All models can be acquired from huggingface.co.

(#)	Model Name	Pre-training	Source
1.	EN-ASR$_{base}$	960h EN Librispeech ASR	http://bit.ly/3wZgfDC
2.	DE-ASR$_{base}$	DE Libri-CV-9 ASR	http://bit.ly/3RA4LzE
3.	XLSR$_{large}$	Cross-lingual data	http://bit.ly/3JNK2qe
4.	DE-ASR$_{large}$	DE Common Voice 6.1. ASR	http://bit.ly/3X5ry7O

Appendix D. Label distribution of filtered datasets

Table D.10
Distribution of annotations of 3 second segments in the filtered version of the SEP-28k-E and FluencyBank datasets used in this paper (Schuller et al., 2022).

SEP-C: Label distribution (% (#))				
class	train	dev	test	Σ
Block	16.43 (1664)	16.66 (744)	15.98 (751)	16.37 (3159)
Interjections (Fillers)	2.99 (303)	2.13 (95)	2.51 (118)	2.67 (516)
Garbage	2.18 (221)	1.97 (88)	1.00 (47)	1.85 (356)
Prolongation	7.64 (774)	8.06 (360)	8.47 (398)	7.94 (1532)
Sound Repetition	6.02 (610)	7.97 (356)	4.89 (230)	6.20 (1196)
Word/Phrase Repetition	8.63 (874)	7.55 (337)	9.17 (431)	8.51 (1642)
No dysfluencies	56.10 (5682)	55.66 (2485)	57.97 (2724)	56.45 (10891)
Σ	52.5 (10128)	23.14 (4465)	24.36 (4699)	100.00 (19292)
FluencyBank-C: Label distribution (% (#))				
class	train	dev	test	Σ
Block	17.10 (280)	17.81 (117)	11.82 (61)	16.30 (458)
Interjections (Fillers)	2.99 (49)	2.28 (15)	5.04 (26)	3.20 (90)
Garbage	2.02 (33)	1.22 (8)	0.97 (5)	1.64 (46)
Prolongation	5.62 (92)	9.74 (64)	10.27 (53)	7.44 (209)
Sound Repetition	9.59 (157)	10.96 (72)	9.50 (49)	9.89 (278)
Word/Phrase Repetition	8.98 (147)	4.11 (27)	5.43 (28)	7.19 (202)
No dysfluencies	53.70 (879)	53.88 (354)	56.98 (294)	54.34 (1527)
Σ	58.26 (1637)	23.38 (657)	18.36 (516)	100.00 (2810)

References

Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., Vijayanarasimhan, S., 2016. YouTube-8M: A large-scale video classification benchmark. CoRR arXiv:1609.08675.

Baevski, A., Hsu, W.-N., Conneau, A., Auli, M., 2021. Unsupervised speech recognition. arXiv:2105.11084 [Cs, Eess].

Baevski, A., Zhou, Y., Mohamed, A., Auli, M., 2020. Wav2vec 2.0: A framework for self-supervised learning of speech representations. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (Eds.), Advances in Neural Information Processing Systems. 33, Curran Associates, Inc., pp. 12449–12460.

Bayerl, S.P., Hönig, F., Reister, J., Riedhammer, K., 2020. Towards automated assessment of stuttering and stuttering therapy. In: Sojka, P., Kopeček, I., Pala, K., Horák, A. (Eds.), Text, Speech, and Dialogue. 12284, Springer International Publishing, Cham, pp. 386–396. http://dx.doi.org/10.1007/978-3-030-58323-1_42.

Bayerl, S.P., Tammewar, A., Riedhammer, K., Riccardi, G., 2021. Detecting emotion carriers by combining acoustic and lexical representations. In: 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). pp. 31–38. http://dx.doi.org/10.1109/ASRU51503.2021.9687893.

Bayerl, S.P., Wagner, D., Hönig, F., Bocklet, T., Nöth, E., Riedhammer, K., 2022a. Dysfluencies seldom come alone – Detection as a multi-label problem. arXiv:2210.15982.

Bayerl, S.P., Wagner, D., Noeth, E., Riedhammer, K., 2022b. Detecting dysfluencies in stuttering therapy using Wav2vec 2.0. In: Proc. Interspeech 2022. ISCA, pp. 2868–2872. http://dx.doi.org/10.21437/Interspeech.2022-10908.

Bayerl, S.P., Wagner, D., Nöth, E., Bocklet, T., Riedhammer, K., 2022c. The Influence of dataset partitioning on dysfluency detection systems. In: Sojka, P., Kopeček, I., Pala, K., Horák, A. (Eds.), Text, Speech, and Dialogue. Springer International Publishing.

Bayerl, S.P., Wolff von Gudenberg, A., Hönig, F., Noeth, E., Riedhammer, K., 2022. KSoF: The Kassel State of Fluency dataset – A therapy centered dataset of stuttering. In: Proceedings of the Language Resources and Evaluation Conference. European Language Resources Association, Marseille, France, pp. 1780–1787.

Belz, M., 2021. Die Phonetik von äh und ähm: Akustische Variation von Füllpartikeln im Deutschen. Springer Berlin Heidelberg, Berlin, Heidelberg, http://dx.doi.org/10.1007/978-3-662-62812-6.

Bernstein Ratner, N., MacWhinney, B., 2018. Fluency Bank: A new resource for fluency research and practice. J. Fluency Disord. 56, 69–80. http://dx.doi.org/10.1016/j.jfludis.2018.03.002.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. In: Information Science and Statistics, Springer, New York.

Boser, B.E., Guyon, I.M., Vapnik, V.N., 1992. A training algorithm for optimal margin classifiers. In: Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory. ACM Press, pp. 144–152.

Cai, X., Yuan, J., Zheng, R., Huang, L., Church, K., 2021. Speech emotion recognition with multi-task learning. In: Interspeech 2021. ISCA, pp. 4508–4512. http://dx.doi.org/10.21437/Interspeech.2021-1852.

Caruana, R., 1998. Multitask Learning. Springer.

Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. SMOTE: Synthetic minority over-sampling technique. J. Artificial Intelligence Res. 16, 321–357. http://dx.doi.org/10.1613/jair.953.

Chung, J.S., Nagrani, A., Zisserman, A., 2018. VoxCeleb2: Deep speaker recognition. arXiv:1806.05622 [Cs, Eess].

Craig, A., Hancock, K., Tran, Y., Craig, M., Peters, K., 2002. Epidemiology of stuttering in the community across the entire life span. J. Speech, Lang. Hear. Res. 45 (6), 1097–1105. http://dx.doi.org/10.1044/1092-4388(2002/088).

Ellis, J.B., Ramig, P.R., 2009. A handbook on stuttering. J. Fluency Disord. 34 (4), 295–299. http://dx.doi.org/10.1016/j.jfludis.2009.10.004.

Esmaili, I., Dabanloo, N.J., Vali, M., 2017. An automatic prolongation detection approach in continuous speech with robustness against speaking rate variations. J. Medical Signals and Sensors 7 (1), 1–7.

Eyben, F., Wöllmer, M., Schuller, B., 2010. Opensmile: The munich versatile and fast open-source audio feature extractor. In: Proceedings of the International Conference on Multimedia - MM '10. ACM Press, Firenze, Italy, p. 1459. http://dx.doi.org/10.1145/1873951.1874246.

Gemmeke, J.F., Ellis, D.P.W., Freedman, D., Jansen, A., Lawrence, W., Moore, R.C., Plakal, M., Ritter, M., 2017. Audio set: An ontology and human-labeled dataset for audio events. In: Proc. IEEE ICASSP 2017. New Orleans, LA.

Gong, Y., Chung, Y.-A., Glass, J., 2021. AST: Audio Spectrogram Transformer. arXiv:2104.01778.

Grósz, T., Porjazovski, D., Getman, Y., Kadiri, S., Kurimo, M., 2022. Wav2vec2-based paralinguistic systems to recognise vocalised emotions and stuttering. In: Proceedings of the 30th ACM International Conference on Multimedia. In: MM '22, Association for Computing Machinery, New York, NY, USA, pp. 7026–7029. http://dx.doi.org/10.1145/3503161.3551572.

Harvill, J., Hasegawa-Johnson, M., Yoo, C.D., 2022. Frame-level stutter detection. In: Proc. Interspeech 2022. pp. 2843–2847. http://dx.doi.org/10.21437/Interspeech.2022-204.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778. http://dx.doi.org/10.1109/CVPR.2016.90.

Hershey, S., Chaudhuri, S., Ellis, D.P.W., Gemmeke, J.F., Jansen, A., Moore, R.C., Plakal, M., Platt, D., Saurous, R.A., Seybold, B., Slaney, M., Weiss, R.J., Wilson, K., 2017. CNN architectures for large-scale audio classification. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, New Orleans, LA, pp. 131–135. http://dx.doi.org/10.1109/ICASSP.2017.7952132.

Hochreiter, S., Schmidhuber, J., 1997. Long Short-Term Memory. Neural Comput. 9 (8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Howell, P., Davis, S., Bartrip, J., 2009. The University College London Archive of Stuttered Speech (UCLASS). J. Speech, Lang. Hear. Res. 52 (2), 556–569. http://dx.doi.org/10.1044/1092-4388(2009/07-0129).

Howell, P., Sackin, S., 1995. Automatic recognition of repetitions and prolongations in stuttered speech. In: Proceedings of the First World Congress on Fluency Disorders. 2, University Press Nijmegen Nijmegen, The Netherlands, pp. 372–374.

Klumpp, P., Arias-Vergara, T., Vásquez-Correa, J.C., Pérez-Toro, P.A., Orozco-Arroyave, J.R., Batliner, A., Nöth, E., 2022. The phonetic footprint of Parkinson's disease. Comput. Speech Lang. 72, 101321. http://dx.doi.org/10.1016/j.csl.2021.101321, arXiv:2112.11514.

Kourkounakis, T., 2021. LibriStutter. http://dx.doi.org/10.5683/SP3/NKVOGQ.

Kourkounakis, T., Hajavi, A., Etemad, A., 2020. Detecting multiple speech disfluencies using a deep residual network with bidirectional long short-term memory. In: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 6089–6093.

Kourkounakis, T., Hajavi, A., Etemad, A., 2021. FluentNet: End-to-end detection of stuttered speech disfluencies with deep learning. IEEE/ACM Trans. Audio, Speech, and Language Processing 29, 2986–2999. http://dx.doi.org/10.1109/TASLP.2021.3110146.

Lea, C., Mitra, V., Joshi, A., Kajarekar, S., Bigham, J.P., 2021. SEP-28k: A dataset for stuttering event detection from podcasts with people who stutter. In: ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, Toronto, ON, Canada, pp. 6798–6802. http://dx.doi.org/10.1109/ICASSP39728.2021.9413520.

Lei, S., Zhang, H., Wang, K., Su, Z., 2019. How training data affect the accuracy and robustness of neural networks for image classification.

Lim, W., Jang, D., Lee, T., 2016. Speech emotion recognition using convolutional and recurrent neural networks. In: 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA). pp. 1–4.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B., 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 10012–10022.

Lustyk, T., Bergl, P., Haderlein, T., Nöth, E., Cmejla, R., 2015. Language-independent method for analysis of German stuttering recordings. In: Interspeech 2015. ISCA, pp. 2947–2951. http://dx.doi.org/10.21437/Interspeech.2015-610.

Montacié, C., Caraty, M.-J., Lackovic, N., 2022. Audio features from the Wav2Vec 2.0 embeddings for the ACM multimedia 2022 stuttering challenge. In: Proceedings of the 30th ACM International Conference on Multimedia. In: MM '22, Association for Computing Machinery, New York, NY, USA, pp. 7195–7199. http://dx.doi.org/10.1145/3503161.3551606.

Noeth, E., Niemann, H., Haderlein, T., Decher, M., Eysholdt, U., Rosanowski, F., Wittenberg, T., 2000. Automatic stuttering recognition using hidden Markov models. In: Sixth International Conference on Spoken Language Processing. 4, pp. 65–68.

Peddinti, V., Povey, D., Khudanpur, S., 2015. A time delay neural network architecture for efficient modeling of long temporal contexts. In: Proc. Interspeech 2015. pp. 3214–3218. http://dx.doi.org/10.21437/Interspeech.2015-647.

Pepino, L., Riera, P., Ferrer, L., 2021. Emotion recognition from speech using Wav2vec 2.0 embeddings. In: Interspeech 2021. ISCA, pp. 3400–3404. http://dx.doi.org/10.21437/Interspeech.2021-703.

Ravanelli, M., Zhong, J., Pascual, S., Swietojanski, P., Monteiro, J., Trmal, J., Bengio, Y., 2020. Multi-task self-supervised learning for robust speech recognition. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, Barcelona, Spain, pp. 6989–6993. http://dx.doi.org/10.1109/ICASSP40776.2020.9053569.

Schuller, B., Batliner, A., Amiriparian, S., Bergler, C., Gerczuk, M., Holz, N., Larrouy-Maestri, P., Bayerl, S., Riedhammer, K., Mallol-Ragolta, A., Pateraki, M., Coppock, H., Kiskin, I., Sinka, M., Roberts, S., 2022. The ACM multimedia 2022 computational paralinguistics challenge: vocalisations, stuttering, activity, & mosquitoes. In: Proceedings of the 30th ACM International Conference on Multimedia. In: MM '22, Association for Computing Machinery, New York, NY, USA, pp. 7120–7124. http://dx.doi.org/10.1145/3503161.3551591.

Settles, B., 2012. Active Learning. In: Synthesis Lectures on Artificial Intelligence and Machine Learning, Springer International Publishing, Cham, http://dx.doi.org/10.1007/978-3-031-01560-1.

Sheikh, S.A., Hirsch, F., Ouni, S., 2022a. Robust Stuttering Detection via Multi-task and Adversarial Learning. In: 2022 30th European Signal Processing Conference (EUSIPCO). p. 5.

Sheikh, S.A., Sahidullah, M., Hirsch, F., Ouni, S., 2021. StutterNet: Stuttering detection using time delay neural network. In: 2021 29th European Signal Processing Conference (EUSIPCO). IEEE, Dublin, Ireland, pp. 426–430. http://dx.doi.org/10.23919/EUSIPCO54536.2021.9616063.

Sheikh, S.A., Sahidullah, M., Hirsch, F., Ouni, S., 2022b. Machine learning for stuttering identification: Review, challenges and future directions. Neurocomputing 514, 385–402. http://dx.doi.org/10.1016/j.neucom.2022.10.015.

Sheikh, S.A., Sahidullah, M., Ouni, S., Hirsch, F., 2022c. End-to-end and self-supervised learning for ComParE 2022 stuttering sub-challenge. In: Proceedings of the 30th ACM International Conference on Multimedia. In: MM '22, Association for Computing Machinery, New York, NY, USA, pp. 7104–7108. http://dx.doi.org/10.1145/3503161.3551588.

Shonibare, O., Tong, X., Ravichandran, V., 2022. Enhancing ASR for stuttered speech with limited data using detect and pass. arXiv:2202.05396.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: Bengio, Y., LeCun, Y. (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., Khudanpur, S., 2018. X-vectors: Robust DNN embeddings for speaker recognition. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 5329–5333. http://dx.doi.org/10.1109/ICASSP.2018.8461375.

Sommer, M., Waltersbacher, A., Schlotmann, A., Schröder, H., Strzelczyk, A., 2021. Prevalence and therapy rates for stuttering, cluttering, and developmental disorders of speech and language: Evaluation of German Health Insurance Data. Front. Hum. Neurosci. 15, 645292. http://dx.doi.org/10.3389/fnhum.2021.645292.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention is all you need. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (Eds.), Advances in Neural Information Processing Systems. 30, Curran Associates, Inc..

Ward, T., Bernier, R., Mukerji, C., Perszyk, D., McPartland, J.C., Johnson, E., Faja, S., Nevers, M., Frazier, T., Howlin, P., Savage, S., Zane, T., Lanner, T., Myers, M., VanBergeijk, E., Huestis, S., Bauminger-Zviely, N., Doehring, P., Voorst, G., Macy, K., Kwon, J.M., McNulty, E., Chapman, S.M., Crowley, M.J., Bean, A., Hyman, S., Scahill, L.D., Wing, L., Catania, A.C., Thorne, J., Kini, U., Moyle, M., Plowgian, C., Happé, F., Case-Smith, J., Schmitt, L., Lewis, M., Weiss, M.J., Gaag, J.V.d.R., Hurst, H., Bonazinga, L., Paul, D.R., Happé, F., Doyle, C.A., McDougle, C.J., Perri, K.S., Moran, M., Stigler, K., McDougle, C.J., Hyman, S., St. John, M., Hessl, D., Schneider, A., Katon, J., Hurst, H., Bauminger-Zviely, N., Hadjikhani, N., Rinehart, N., Enticott, P., Bradshaw, J., Palmieri, M., LaRue, R., Molteni, J., LaRue, R., Palmieri, M., Prelock, P., Müller, R.-A., LaRue, R., Egan, S., Hendricks, D., Holman, K.C., D'Eramo, K., Faja, S., Perszyk, D., 2013. Fluency and fluency disorders. In: Volkmar, F.R. (Ed.), Encyclopedia of Autism Spectrum Disorders. Springer New York, New York, NY, pp. 1308–1310. http://dx.doi.org/10.1007/978-1-4419-1698-3-1931.

Wenninger, M., Bayerl, S.P., Schmidt, J., Riedhammer, K., 2019. Timage–a robust time series classification pipeline. In: International Conference on Artificial Neural Networks. Springer, Cham, pp. 450–461.

Wingate, M.E., 1984. Fluency, disfluency, dysfluency, and stuttering. J. Fluency Disord. 9 (2), 163–168. http://dx.doi.org/10.1016/0094-730X(84)90033-0.

Yairi, E., Ambrose, N., 2013. Epidemiology of stuttering: 21st century advances. J. Fluency Disord. 38 (2), 66–87. http://dx.doi.org/10.1016/j.jfludis.2012.11.002.

You, K., Xu, K., Zhu, B., Feng, M., Feng, D., Liu, B., Gao, T., Ding, B., 2022. Masked modeling-based audio representation for ACM multimedia 2022 computational paralinguistics ChallengE. In: Proceedings of the 30th ACM International Conference on Multimedia. In: MM '22, Association for Computing Machinery, New York, NY, USA, pp. 7060–7064. http://dx.doi.org/10.1145/3503161.3551579.

Zhang, X., Valles, I., Yu, C., Droppo, J., Stolcke, A., Barra-Chicote, R., Ravichandran, V., 2022. Stutter-TTS: Synthetic generation of diverse stuttered voice profiles. In: NeurIPS 2022 Workshop on Synthetic Data for Empowering ML Research.