Assessing Comprehensibility of Children's Read Speech

Submitted in partial fulfillment of the requirements
of the degree of

Dual Degree (B.Tech + M.Tech)

by

Mithilesh Vaidya
(Roll No. 17D070011)

Supervisor:
Prof. Preeti Rao

Department of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY BOMBAY
June 2022

Thesis Approval

This thesis entitled **Assessing Comprehensibility of Children's Read Speech** by **Mithilesh Vaidya** is approved for the degree of **Dual Degree (B.Tech + M.Tech)**.

Examiners:

Sunil
Sunil (Jun 30, 2022 13:12 GMT+5.5)
....

Dr. Sunilkumar Kopparapu

V. RajBabu
Rajbabu (Jun 30, 2022 10:53 GMT+5.5)
....

Prof. Rajbabu Velmurugan

Supervisor:

Preeti Rao
....

Prof. Preeti Rao

Chairperson:

V. RajBabu
Rajbabu (Jun 30, 2022 10:53 GMT+5.5)
....

Prof. Rajbabu Velmurugan

Date: 29/06/2022

Place: Indian Institute of Technology Bombay, Mumbai, India

Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

[illegible]

Date: 19/06/2022

Mithilesh Vaidya

Roll No. 17D070011

Abstract

In this work, we focus on evaluating oral reading fluency from audio recordings of read aloud text. Apart from word decoding accuracy and speaking rate, prosody or expression is an important indicator of the reader's comprehension of text. The goal is to predict a fluency or comprehensibility score for a given speech utterance that is typically a short passage of connected text. It is challenging because the subjectivity of scoring leads to noisy labels due to rater-specific biases. Also, obtaining such ratings from trained experts is an expensive procedure, which prevents the collection of large datasets which can be exploited by standard deep learning models. As a result, majority of the existing literature is focused on carefully designed knowledge-based features which are combined with classification or regression models. A Random Forest Classifier (RFC) trained on various word-level and recording-level aggregates of acoustic features such as pitch, energy and duration and text-based features such as WCPM and lexical miscues serves as the baseline for this work. We first experiment with deep learning alternatives for the above features and observe performance which is comparable to RFC. One of the main goals of this work is to replace this extensive hand-engineering with deep learning architectures to not only reduce the complexity of the pipeline but also potentially improve performance.

Since performance of deep learning architectures scales with the amount of data, models pre-trained on large corpora have proven to be very effective, especially in low-resource scenarios. We experiment with one such pre-trained model called Wav2vec2.0 which has been extensively used for ASR, emotion recognition and other speech tasks. Our simplest architecture, operating purely on raw waveform as input, already exceeds the performance of the RFC trained on both acoustic and lexical features. Unlike the hand-crafted features, wav2vec representations are not interpretable. Hence, to crack open the black box that is wav2vec, we analyse the latent information encoded in the high-level wav2vec representations by probing them for correlation with the hand-crafted features which are interpretable by definition. Several expected and unexpected results are discussed. To exploit the possibly complementary information in HC features, we propose an architecture which supports concatenation of these features at each hierarchy. We find that incorporating non-acoustic recording-level HC features give a slight improvement in performance. Apart from concatenation, we explore two other fusion techniques for combining the wav2vec and HC features. An output representation from a RNN, trained on hand-crafted word-level features, is combined with the wav2vec representation at different levels and a slight improvement in performance is observed, thereby indicating complementary information present in the word-level HC features. Another approach for exploiting information in HC features is multi-task learning (MTL). Along with comprehensibility rating, we tune the model to simultaneously predict these HC features. This nudges the model to learn intermediate representations which capture such features, which are known to be useful for comprehensibility. The key benefit of such a framework over concatenation is that we do not require the HC features during inference. Although we did not observe any improvement in performance, we believe it is a powerful scalable framework which deserves more extensive experimentation. We summarise our work in the final chapter and propose several interesting directions to pursue in the future, including an extensive discussion on a promising self-supervised learning framework that can exploit the large amounts of unlabelled data available.

Contents

List of Tables

List of Figures

List of Abbreviations

HC	Hand-crafted
MTL	Multi Task Learning
RFC	Random Forest Classifier
RFECV	Recursive Feature Elimination using Cross Validation
PED	Prosodic Event Detection
SSL	Self-supervised Learning
WCPM	Words Correct Per Minute

Chapter 1

Introduction

The ability to read is one of the most fundamental skills [7]. Most people develop this skill in school. Since it forms the foundation for acquiring knowledge in today's world, children who are poor readers suffer extensively in school and everyday life. They are slow to acquire knowledge and this has a cascading effect on their ability to interact with the world [8].

Reading ability can be assessed in various ways such as conducting exams based on a written text. However, it places additional cognitive demands on the child to read and understand the questions. Hence, we focus on the automatic assessment of oral reading fluency, which is considered to be an accurate methodology for determining a child's reading ability [7]. Children are asked to read out a known text (also called canonical text) and their reading ability can be assessed by analysing the fluency of the spoken utterance [9].[1] Here, fluency refers to accuracy, automaticity[2], and oral reading prosody, which, taken together, facilitate the reader's construction of meaning.

Reading can be divided into two components: word decoding and meaning construction.

[1]Assuming the child has no speaking disorder such as stammer.

[2]Automaticity is the ability to do things without occupying the mind, allowing it to become an automatic response pattern or habit. It comes with practice.

Word decoding ability can be judged by combining the speech rate (measured in words per minute) with accuracy (number of words read correctly) to obtain a metric called WCPM (words correct per minute) [10]. Readers with poor WCPM struggle to decode and pronounce the words at a reasonable rate. Since word decoding ability is the first stage in comprehending text, such readers can be immediately flagged with poor reading comprehension ability.

Once a child attains a certain level of word decoding proficiency, prosodic aspects start gaining more importance [11]. In order to comprehend the text and make it comprehensible to a listener, one needs to process it at a higher level; simply reading out the words is not sufficient. This includes identification of words, relating them to words in the neighbourhood by taking the sentence structure and syntax into account and updating one's internal mental model. When reading text aloud, we modulate our voice to reflect this internal process. It is acoustically correlated with prosodic aspects of speech such as rhythm, pausing, intonation and stress. Proficient readers tend to convey the linguistic focus and the author's sentiments using prosodic variations. As a result, a simple ASR-based framework which compares the ASR-decoded text with the canonical text will be an unreliable indicator of oral reading ability since it fails to capture the all important prosodic aspect. Thus, high WCPM is a necessary but not a sufficient criterion for reading proficiency.

In stage 1 of this dual-degree project, we presented a deep learning solution for the sub-task of prosodic event detection (PED). We focused on two prosodic events: prominence and phrase boundary. Prominence refers to the phenomenon of stressing on a particular linguistic unit such as word/syllable/etc. in order to convey novel information. It helps the listener focus on a particular part of the utterance. Phrase boundary (phrasing) refers to the grouping of words in order to convey syntactic information (analogous to punctuation in text). The two phenomenon are important indicators of oral reading ability [12]. Our proposed system, which outperformed a random forest classifier operating on hand-crafted features, was tested on manually obtained prosodic events labels on our read speech dataset. Please see section 1.1 for more details.

A framework which can reliably assess the reading skills of children at scale has the potential for massive social impact. The identification of poor readers can help attract special attention to them in terms of pedagogic interventions to improve their reading skills. One must also keep the scale of this problem in mind while designing models; any human involvement in the reading assessment pipeline can be a bottleneck in the overall system.

In a nutshell, a human expert (such as a language teacher) can reliably judge the reading ability of an individual by asking him/her to read out a short paragraph or story and analysing the recorded utterance. This task of rating an utterance on a scale of 0-5, taking into account both word decoding ability and fluency, is referred to as comprehensibility rating.

The current work builds upon the recent Ph.D. thesis of Kamini Sabu [1] where a large set of hand-crafted features was derived from the waveform, ASR decoded output and the canonical text for the comprehensibility prediction of read text comprising several sentences. Acoustic features include functionals of pitch, intensity and energy at various hierarchies (word-level and recording-level) while lexical features include word-level POS tags, miscue features, etc. and recording-level features such as Words Correct Per Minute (WCPM) and prosodic miscue aggregates. Recursive Feature Elimination using Cross Validation (RFECV) is used in tandem with a Random Forest Classifier (RFC) to derive a compact set of features of highly relevant features. The best system, operating on features derived from the waveform and manually transcribed text, reported a Pearson corr. of 0.777. However, such a system has a few drawbacks. Firstly, the feature extraction stage has multiple independent components which cannot be jointly optimised for the task of comprehensibility prediction. Secondly, extracting such features requires significant domain knowledge. Thirdly, since the feature space is very large, we might miss out on important features or patterns in the data when using hand-engineering.

With the advent of deep learning, end-to-end models trained on vast datasets have outperformed traditional methods in almost every domain. Their key promise is the ability to learn powerful representations directly from data. Backpropagation enables joint optimisation of various components of the system in an end-to-end fashion. Minimal hand-engineering ensures that the most relevant features are automatically learned from raw data.

Zhou et al. [13] (of Educational Testing Services) proposed a deep learning model for fluency assessment of non-native English speakers. The last time step output of a bidirectional LSTM, operating on frame-level contours (such as F0, loudness, MFCCs, etc.) is extracted. It is concatenated with utterance-level hand-crafted features (such as fluency, grammar and vocabulary use) and passed through a simple linear layer to predict a 4-point score for 45 seconds of spontaneous speech. They observed an improvement of around 1% improvement over the use of hand-crafted features. An improved architecture was subsequently proposed by the same group in [14]. Word-level acoustic features (such as mean pitch) and lexical features (GloVe embeddings of ASR hypothesis) are input to feature extractors such as CNNs and attention-augmented RNNs. The output of each modality is concatenated and passed through a linear regression layer to predict the score. They observed that BLSTM, coupled with an attention layer, significantly improved performance. A similar approach is adopted by [15]. Outputs of the acoustic branch, consisting of a CRNN operating on spectrograms, and a BLSTM, operating on GLoVe embeddings, are fused with a novel attention fusion network. The attention weights can be interpreted as the importance of each modality in predicting the final score. Lastly, separate BLSTMs are proposed in [16] for modelling Delivery, Language use, and Content in an automated speech scoring system. Each response in the dialogue is rated separately by the three sequential models. The final speech proficiency score is obtained by fusing the three subscores. Their proposed system attained a Pearson corr. of 0.747, an improvement of 0.063 over the RFC baseline operating on hand-crafted speech features [17]. Note that all the above systems operate on some set of hand-crafted features, be it at the frame, word or recording level.

In this work, we propose a deep learning framework for automatic reading fluency assessment. Our model is based on a recently proposed self-supervised model called Wav2vec2.0 [6] (referred to as wav2vec herafter). Pre-trained on a large unlabelled speech corpus in a self-supervised fashion, wav2vec generates a robust frame-level representation of speech. In this work, we use the pre-trained model to extract frame-level features and experiment with various deep learning modules on top to predict the comprehensibility rating. The simplest model, which extracts wav2vec embeddings from waveform, transforms them using a stack of fully-connected layers and uses mean pooling to obtain a recording-level representation, was already found to outperform a RFC trained on both lexical and acoustic features. This encouraging result motivated us to stick to a wav2vec backbone.

A major drawback of deep learning systems is their black-box nature. Unlike hand-crafted features, which are interpretable by definition, deep learning representations are opaque. Despite superior performance compared to traditional techniques, it is difficult to reason why a deep learning model predicted a given score. This presents a major challenge for any attempts to improve the model by bringing in complementary information. Recently, works such as [18] have attempted to tackle this issue by calculating the effect of tweaking a particular input feature to understand it's effect on the final output. On the other hand, in [19], the authors analyse the acoustic feature representations of a CNN trained for prosodic event detection by regressing HC features such as duration and energy. We propose a similar methodology for wav2vec representations. Linear probes consisting of fully-connected layers are added on top of representations (derived from wav2vec) at various hierarchies to predict the HC features. On inspecting the Pearson correlation of the probe output and the HC features, we can draw conclusions regarding the nature of information captured by the wav2vec representations.

Various studies have demonstrated the effectiveness of using both acoustic and lexical information for the task of speech emotion recognition [20, 21, 22]. Similarly, for comprehensibility prediction, we observe an improvement over the purely acoustic model on incorporating various lexical features [1].

To inspect if such hand-crafted features contain any additional information, we concatenate them at various hierarchies (word-level and recording-level) to check for any improvement in performance. These are not limited to lexical features. For example, in [2] low-level descriptors (frame-level features such as pitch, jitter, formant energies, etc.) improve performance on concatentaion with wav2vec embeddings. In our case, we see a slight improvement on using non-acoustic recording-level features. Simple concatenation of lexical features with word-level wav2vec representations did not improve performance. Hence, we propose two alternate techniques for fusing the lexical and acoustic modalities. A marginal improvement is observed on introducing a parallel RNN branch operating on word-level HC features (along with the original wav2vec branch).

By combining the insights from the feature concatenation experiments and probing experiments, we propose a multi-task learning framework for learning better representations. By adding auxiliary tasks for predicting HC features at various hierarchies and training the model end-to-end with comprehensibility rating as the primary task, we nudge the model to learn better internal representations. However, we did not observe any improvement in performance and we discuss a few possible reasons for the same.

Lastly, we summarise our work and present the best performance of each of the proposed systems, along with the baseline. We then propose a few interesting directions to pursue, along with an extensive discussion on a promising self-supervised model which can exploit unlabelled data.

1.1 Summary of Stage 1

For the task of prosodic event detection, a random forest classifier operated on a large set of hand-crafted features which were derived on incorporating extensive domain knowledge [23]. In stage 1 of our work, we replaced this system with a convolutional recurrent neural network (CRNN) operating directly on segmented speech waveforms. Sinc-based filtering, along with multi-task learning for exploiting the linguistic association between prominence and phrase boundary, helped the model outperform a GRU operating on hand-crafted features. For more details, please refer to our work which was presented at ICASSP 2022 [3].

We used this system to detect word-level prosodic events. Word-level prosodic miscue tags and recording-level aggregates of prosodic miscues such as F-score, precision, recall, etc. were computed by comparing the predictions with the canonical text information structure (whether a word should have been prominent or phrase break). As discussed in the introduction, these are reliable indicators of oral fluency. Subsequently, we tried out a wav2vec-based architecture for the task of PED. We observed performance superior to the Sinc+MTL model. More details regarding the performance of wav2vec for PED can be found in Appendix A.

1.2 Dataset

The dataset used in this work is identical to that used in [1]. We review the salient characteristics of our dataset here. Our dataset consists of recordings collected from various schools in Maharashtra. All students belong to the age group of 10-14 years with English as the second language studied in school. We apply two filters in order to obtain the final dataset: firstly, all recordings have a WCPM (words correct per minute) of over 70. Secondly, we only consider recordings with a lexical miscue rate which is below 15% (lexical miscue rate refers to the total number of word-level omissions, insertions and substitutions in the manually transcribed text with reference to the canonical text). We apply these two filters since recordings which do not meet either criteria are considered incomprehensible due to low accuracy and rate.On applying these criteria, we have 1447 recordings read by 165 students from a pool of 148 unique paragraph texts. Each paragraph has 50-70 words. The duration of the recordings varies from 12

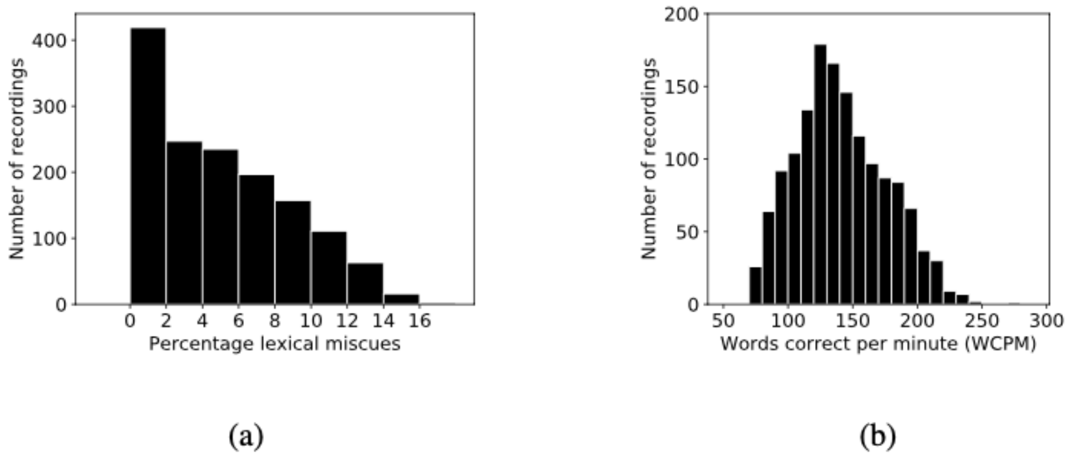


Figure 1.1: Distribution of (a) % lexical miscues (b) WCPM across the set of 1447 recordings.

seconds to 56 seconds with a mean recording duration of 25 seconds and standard deviation of 8 seconds. The total duration of the dataset is approximately 10 hours. The distribution of WCPM and % lexical miscue is given in figure 1.1. Both parameters are (moderately) uniformly distributed.

The distribution of the number of recordings per speaker is given in figure 1.2. We see that the final set of recordings contains more than 50 recordings for a couple of speakers, while most speakers have less than ten recordings.

1.2.1 Comprehensibility ratings

Each recording is independently rated by two English language teachers on a scale of 0-5. The rating rubric given in table 1.1 is similar to the one prescribed by NAEP [24].

As seen from the histogram in figure 1.3, the two raters have a different approach; rater 1 utilises the entire 6-point scale while rater 2 tends to focus on the 1, 2, 3, 4 range. Another way to visualise this difference is by plotting the distribution of the ratings across the two raters (figure 1.4). A non-diagonal plot is an indicator of rater-specific bias.

To remove this bias, we Z-score normalise the ratings of each rater separately and take the

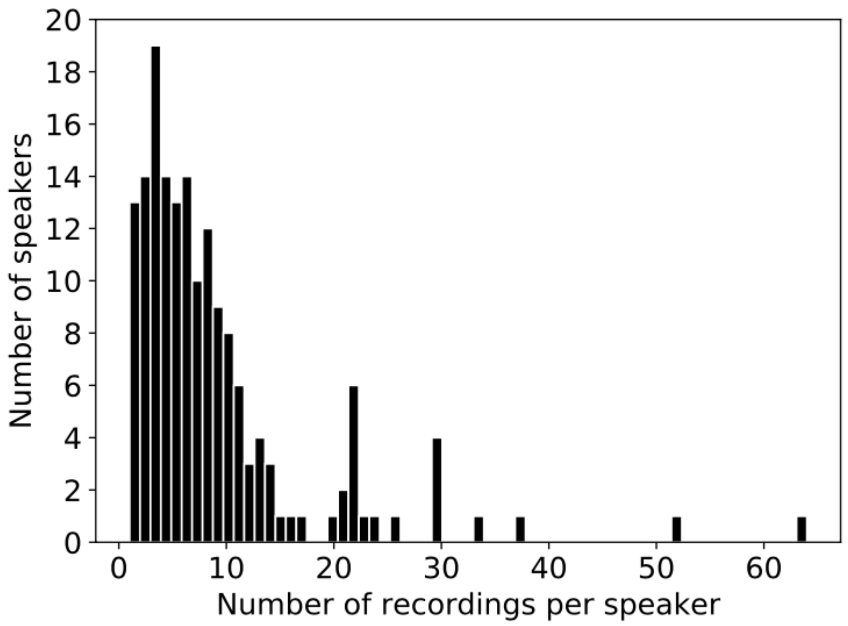


Figure 1.2: Distribution of number of recordings per speaker.

Rating level	Interpretation
0	random grouping, pauses in wrong places, incomprehensible to the listener
1	poor grouping by and large with glimpse of good grouping in one or two places. The underlying reason could be word difficulty.
2	better grouping but still not perfect in all places.
3	grouping is good but stresses are wrong.
4	basically a good reader, some local slips that may be attributed to first-time reading of the given text. This level can be treated as the benchmark.
5	exceptional (could be a practiced child who does speech/elocution training).
Can't rate	reserved for unusual cases such as the audio does not contain the child's speech, volume is too low, too noisy, too short, etc.

Table 1.1: Rating rubric used by the raters to assign a score between 0 and 5 to a given recording.

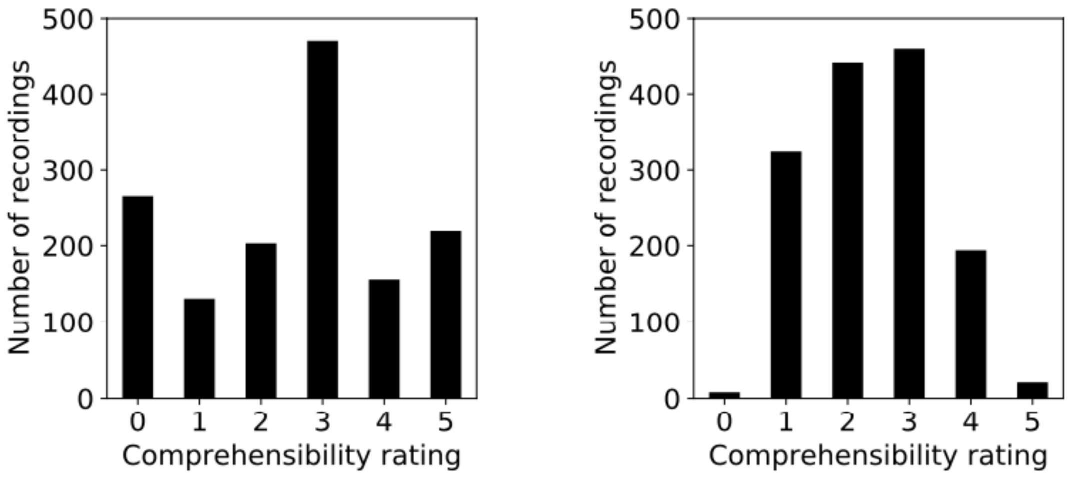


Figure 1.3: Histogram of distributions by the two raters. While rater 1 (left) tends to use the entire 6-point scale, rater 2 (right) mainly assigns ratings from the set {1, 2, 3, 4} and rarely chooses the extremes (0 and 5).

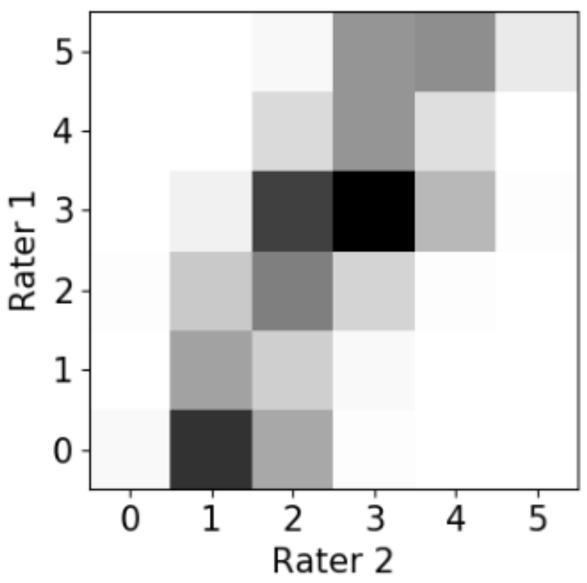


Figure 1.4: Distribution of ratings across the two raters.

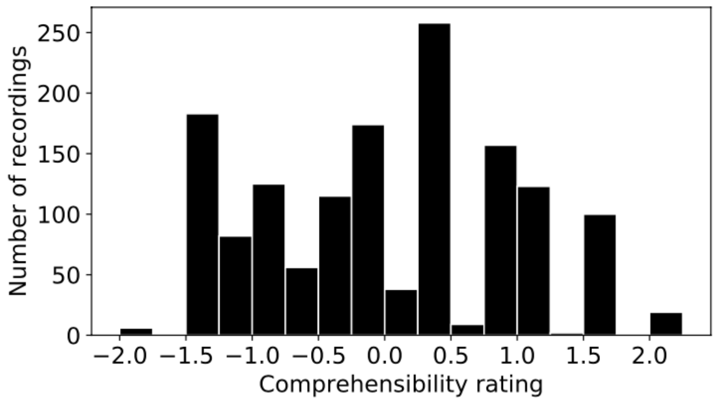


Figure 1.5: Distribution of ratings on averaging the two scores obtained after rater-specific z-score normalisation.

mean of the ratings as the final label. In other words, for a given utterance,

$$y_{mean} = (\frac{y_1 - \mu_1}{\sigma_1} + \frac{y_2 - \mu_2}{\sigma_2})/2 \tag{1.1}$$

where y_1 is the original (integer) rating by rater 1 and μ_1 and σ_1 are the mean and standard deviation respectively of the entire set of ratings by rater 1 (similar terminology for rater 2). y_{mean} is the final averaged rating. Note that $y_{mean} \in \mathbb{R}$ while $y_1, y_2 \in \mathbb{Z}$. For training, we use min-max normalisation to ensure that the labels y_{mean} belong to the range [0, 1]:

$$y_{train} = \frac{y_{mean} - min}{max - min} \tag{1.2}$$

where min = -1.9 and max = 3.91 are obtained from the set y_{mean} from the set of 1447 ratings.

The final averaged continuous rating distribution is given in figure 1.5. By removing the rater bias and providing a richer set of continuous targets as ground truth instead of rounding them up to integers, we expect the model to learn fine-grained differences between utterances.

To ensure that the final predictions are interpretable on the original 6-point scale, we perform reverse normalisation as mentioned in [1]. To do so, we compute the mean (μ_{both}) and standard deviation (σ_{both}) of the entire rating set consisting of ratings from both speakers. Then, during inference, the model prediction which lies between 0 and 1 is first reverse-normalised to undo the min-max normalisation:

$$y^{'} = y * (max - min) + min \tag{1.3}$$

where y is the model prediction, min = -1.9, max = 3.91 (from equation 1.2) and $y^{'}$ is the rating after removing min-max normalisation. Then, we undo the rater-bias normalisation:

$$y_{final} = y^{'} * \sigma_{both} + \mu_{both} \tag{1.4}$$

where y_{final} is the final continuous prediction. Note that it need not belong to the range [0, 5]. Values outside this range are clipped to [0, 5].

1.2.2 ASR and manually transcribed data

As discussed previously, lexical fluency is an important factor which contributes to the comprehensibility score. The spoken words can be compared with the intended (canonical) text that the student was supposed to read. This comparison yields lexical errors made by the student, which is an important indicator of lexical fluency. We need an Automatic Speech Recognition (ASR) system to achieve this. Moreover, ASR also provides the word and phone alignments that help compute acoustic features across different entities like phone, syllable, word, and word group.

The ASR system consists of two components: acoustic model and language model. A Time Delay Neural Network (TDNN) is used as the acoustic backbone [25]. It is trained on 80 hours of Indian English Adult Speech made available through the IITM English ASR challenge by the Speech Processing Lab of IIT Madras. The model is further adapted on 30.5 hrs of children's English reading data by selective fine-tuning of layers. A tri-gram language model trained on canonical story texts read by children forms the second component of the entire system. For more details, please see section 5.2 of [1]. The word error rate of the ASR system on our comprehensibility dataset is 5.55% (section 5.3 of [1]).

Two versions of hand-crafted features are obtained:

- ASR: The ASR-decoded text and the subsequent word alignments are used for generating all recording and word-level acoustic and lexical features.

- FA: The recordings are manually transcribed and then aligned to the manual transcript. The hand-crafted features are subsequently extracted.

The ASR and FA dataset mainly differ in the non-acoustic features. For example, lexical miscue features such as number of miscues, hesitations, etc. will be more accurate for the FA dataset since the ASR-decoded text will be error-prone.

1.3 Ensembling

We use ensembling to reduce model bias and improve performance. The dataset is split into 6 speaker non-overlapping folds. One fold is left out as the test set. Among the remaining 5 folds, we train 5 models in a cross-validation manner i.e. one fold is further left out as the validation set while one model is trained on the rest of the four folds. The validation performance on this left out fold is used for early stopping while mean validation performance across all the models is used for hyperparameters tuning. Refer to tables 1.2 and table 1.3 for a detailed split of the folds.

To generate results on the test set, we average out the predictions of the 5 models. For example, let O_y^x denote the output of model x.y on test fold x when fold y is left out for validation and the rest are used for training.
Final prediction on test fold 1 = $(O_1^1 + O_2^1 + O_3^1 + O_4^1 + O_5^1)/5$.

Similarly, for test fold 2, we have:
Final prediction on test fold 2 = $(O_1^2 + O_2^2 + O_3^2 + O_4^2 + O_5^2)/5$.

Model name	Train folds	Validation fold (for early stop)	Test fold
1.1	3,4,5,6	2	1
1.2	4,5,6,2	3	1
1.3	5,6,2,3	4	1
1.4	6,2,3,4	5	1
1.5	2,3,4,5	6	1

Table 1.2: List of train and validation folds when fold 1 is left out for testing.

Model name	Train folds	Validation fold (for early stop)	Test fold
2.1	3,4,5,6	1	2
2.2	4,5,6,1	3	2
2.3	5,6,1,3	4	2
2.4	6,1,3,4	5	2
2.5	1,3,4,5	6	2

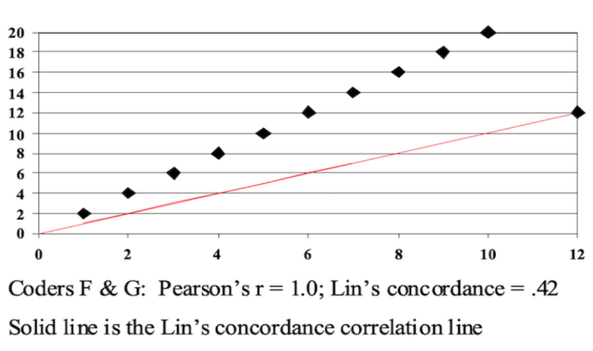
Table 1.3: List of train and validation folds when fold 2 is left out for testing.

1.4 Evaluation metrics

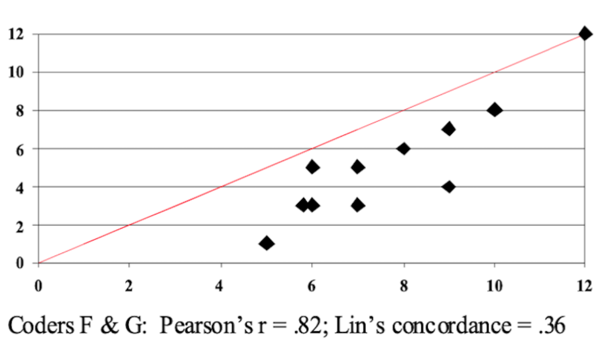
For tasks which involve the prediction of continuous values, Root Mean Square Error (RMSE) and Pearson correlation are two common metrics which are used to evaluate the performance of a system. We discuss the benefits and drawbacks for both metrics and propose to instead track the performance of our models using a metric called Concordance Correlation Coefficient.

Pearson correlation is a common metric for tracking the performance of a regression model. It is designed to capture a linear relationship between two variables and hence is invariant to scale and bias. If X_1^T and Y_1^T are two time series of length T with Pearson correlation ρ, $aX_1^T + b$ (where a and b are scalars) and y will also have a Pearson correlation of ρ. If we consider Y_1^T to be the ground truth comprehensibility ratings and X_1^T to be the model predictions, Pearson correlation is not a sufficient metric.[3] This is because in order to deliver a final score,

[3]In principle, it could be possible to use Pearson correlation as a metric, along with a small validation dataset to correct for scale and bias. However, it will complicate the procedure of choosing the best models.



(a) Example 1



(b) Example 2

Figure 1.6: In example 1, Pearson corr. is high despite the difference in the scale of the two variables. In example 2, Pearson corr. is high despite the difference in the bias of the two variables. In both cases, Lin's CCC is low since it penalises differences in both bias and scale. Figures reproduced from [4].

we want the predictions to match the interpretable rating scale which is used for labelling the utterances. In other words, it is desired that the model predictions are scale-free and bias-free.

Although RMSE is a natural choice to alleviate this issue, it is sensitive to outliers. Moreover, it is dependent on the scale and bias of the rating scale and hence comparing performance across datasets which use different rating scales is not possible. Competitions for related tasks such as speech emotion recognition now use a metric called Concordance Correlation Coefficient (CCC) for evaluating models [26].

CCC is a combination of Mean Squared Error (MSE) and Pearson correlation. For two time series x and y, CCC is defined as:

$$CCC(x,y) = \frac{2\rho_{xy}\sigma_x\sigma_y}{\sigma_x^2+\sigma_y^2+(\mu_x-\mu_y)^2} \tag{1.5}$$

where ρ_{xy} is the Pearson correlation between the two series, σ_x and σ_y are the standard deviations of the two series and μ_x and μ_y are the corresponding means. The σ terms penalise the model when the scales of the two series are mismatched, the μ terms penalise the model for the bias while ρ captures the correlation. It is bounded between [0, 1] and is hence highly interpretable; higher the value, higher the performance.

For the aforementioned reasons, we use CCC [27] as the main evaluation metric. We report the Validation CCC, Test CCC and Test Pearson[4]. Mean and standard deviation for test is calculated across the 6 test folds while for validation, it is reported across all 30 trained models.

1.5 Outline of report

The report is organised into 7 chapters and 3 appendixes as described below.

In Chapter 1, we introduce the problem, discuss the dataset, our training methodology, evaluation metric for reporting performance and summarise our contributions.

In Chapter 2, the baseline model and the hand-crafted features on which it operates is discussed. We then propose deep learning alternatives which operate on the same set of hand-crafted features. Also, we report important training details such as loss functions and procedure for hyperparameter tuning. Lastly, results for the above models are presented.

In Chapter 3, we propose our first Wav2vec2.0-based architecture. We further discuss a tweaked version and the choice of layers from which wav2vec representations are extracted. Results are subsequently discussed.

In Chapter 4, we discuss a technique for interpreting the information encoded in the latent wav2vec representations. We probe for known hand-crafted features which led to some interesting observations.

In Chapter 5, we present two architectures for concatenating the hand-crafted feature sets at frame, word and recording-level. We then discuss the performance of both models. Our best performing system can be found in this chapter.

In Chapter 6, we propose a multi-task learning framework for learning more robust representations by exploiting the hand-crafted features in an auxiliary branch.

In Chapter 7, we summarise our work, list the performances of every architecture discussed in this work and propose a few directions to explore in the future, along with a promising

[4]We still report Test Pearson since it is a common evaluation metric for regression tasks.

self-supervised framework.

In Appendix A, we discuss a wav2vec-based architecture for prosodic event detection (PED), a potential replacement for the PED work done in stage 1.

In Appendix B, implementation issues for pooling frame-level representations to obtain word-level representations is discussed.

In Appendix C, a preliminary analysis of the time taken by W2Vanilla is presented.

1.6 Contributions

- Our work is the first to propose an end-to-end trained waveform-based model for the task of oral reading fluency assessment which does not involve any feature extraction, thereby significantly simplifying the overall pipeline by learning the features entirely from the data.
- Unlike other approaches which train a model from scratch, our architecture is built on top a pre-trained model and hence we believe it can provide very good performance in data-scarce scenarios too.
- We demonstrate a deep learning framework that outperforms the baseline RFC model, based on hand-crafted features, by a large margin (an improvement of **0.06** absolute in Pearson correlation)
- With the help of probes, we examine the nature of information which is being captured by the black-box Wav2vec2.0-based model tuned for comprehensibility.
- We investigate potentially complementary information in the previously proposed knowledge-based hand-crafted features via fusion.
- Apart from comprehensibility, we attempt to predict a rich set of knowledge-based hand-crafted features at word-level and recording-level in a multi-task learning (MTL) framework to make the representations more robust.
- Finally, we summarise the performance of our systems, propose a self-supervised learning framework and provide other interesting directions for future work.

Chapter 2

Baseline systems with hand-crafted features

In this chapter, we present various baseline models which operate on hand-crafted (HC) features developed in the previous work [1]. These features are extracted from a combination of the input waveform, ASR output and the canonical text. Our goal is to eventually replace such extensive hand-engineering with end-to-end deep learning models.

2.1 Hand-crafted features

The HC features used in this work were derived in [1] for the task of comprehensibility prediction. They capture aspects such as lexical accuracy, speaking rate and prosody and are briefly described next. A TDNN-based ASR model, along with a language model for Indian English, is used to obtain ASR-decoded text. Frame-level contours such as pitch, energy, intensity, etc. were extracted from the waveform using traditional signal processing techniques. Using word boundaries obtained from ASR alignments, these contours are aggregated at the word-level and the recording-level using functionals such as mean, max, standard deviation, etc. On the other hand, the ASR-decoded text (or manual transcripts) are used for obtaining various word identity features such as POS tags. On comparing the decoded text with the canonical text, we can obtain miscue tags for each word. They are also aggregated at the recording-level to obtain high-level lexical features such as number of insertions, deletions, hesitations, etc. Silence, pause and duration features at the syllable and word level are obtained from a combination of acoustic and lexical information. In stage 1, we presented a waveform-based model for Prosodic Event Detection. This model is used for obtaining prominence and phrase boundary predictions for each word in the corpus (check section 1.1 for more details). On comparing them with the information structure, we obtain prosodic miscue tags for each word. We then aggregate it at the recording-level using measures such as precision, recall and F-score.

A list of such features according to their hierarchy and a brief description of the same is given in table 2.1. Refer to [1] for more information. We train deep learning models on these sets of hand-crafted features.

A compact set is further derived using Recursive Feature Elimination Cross Validation (RFECV). A Random Forest Classifier (RFC) trained on this compact feature set serves as our first baseline[1]. A schematic of a RFC is given in figure 2.1.

2.2 Neural network classifiers

In this section, we simply replace the RFC with neural network alternatives which operate on the same set of hand-crafted features.

2.2.1 Recording-level models

We experiment with a simple MLP-based architecture which operates on recording-level features. Various feature sets are passed through a stack of fully-connected (FC) layers (with dropout for regularisation and PReLU as activation function[2]). The FC stack is similar to the

[1]We train the deep learning models on the entire set instead of this compact set; the model should ideally learn to ignore features which are not helpful.

[2]Since it degraded performance, we skipped BatchNorm layers in the stack. A potential explanation is the shallowness of our FC stacks - BatchNorm is helpful for stabilising training of very deep networks and hence may

Hierarchy	Feature set	# features	Examples
Recording	Lexical miscues	8	# insertions, deletions, hesitations
	A-P contour and Pause	88	Functionals of pause, aggregates of word pitch properties across recording
	Prosodic miscue	46	Precision and Recall of word-level prominence and phrase boundary
	Rate	5	WCPM, speed variation
Word	A-P contour (Acoustic)	29	Various band intensities, pitch contour correlation with various shapes
	Duration	11	Functionals of silence and syllable durations
	Lexical identity (LexId)	16	POS tags, content word flag
	Lexical miscues (LexMisc)	4	Binary flags C/I/S for correct/inserted/ substituted, noInsertions before
	Prosodic miscues (PEDMisc)	6	Information structure, automatic prosodic event predictions obtained from stage 1, Prosodic miscue
Frame	Acoustic	19	Pitch, Energy, Intensity, HNR, Auto-correlation

Table 2.1: List of hand-crafted features derived in [1] for comprehensibility prediction. Note that RFC uses a subset of these features for final prediction, as chosen by the RFECV mechanism. A few examples from each feature set are listed in the column *Examples*.

Pointwise Conv1D module used in [2]. The final layer consists of a single output neuron which, on Sigmoid activation, is squished to the range (0, 1). This denotes the final comprehensibility score prediction. The architecture is given in figure 2.2(a).

2.2.2 Word-level models

To obtain a recording-level score from word-level features, we need a sequential model which can accept a variable-length input and extract patterns from the sequence. For simplicity, we use a simple Recurrent Neural Network (RNN) for this purpose. At every time step, a feature vector corresponding to each word in the utterance is fed to the RNN. This vector can include any combination of word-level features mentioned in table 2.1. The output of the RNN is pooled using mean pooling, max pooling or by extracting the last time step output to get a recording-level representation. The final representation is passed through a stack of FC layers to obtain the comprehensibility score. Refer to figure 2.2(b) for the architecture.

After tuning the hyperparameters, we use max pooling of the RNN outputs as the pooling mechanism since it gave the best results. A 1-layer bidirectional GRU with 250 hidden units is used as the sequential model. The max pooled output is passed through a fully-connected layer with 300 hidden units, followed by a single output neuron to obtain the final comprehensibility prediction.

2.3 Training details

In this section, we discuss some crucial aspects of training the neural network models. These include choice of loss function, procedure for hyperparameter tuning, the optimiser used for computing gradients, etc. Note that the discussion is applicable to not just the baseline models but all subsequent models proposed in this work.

not be required in our case

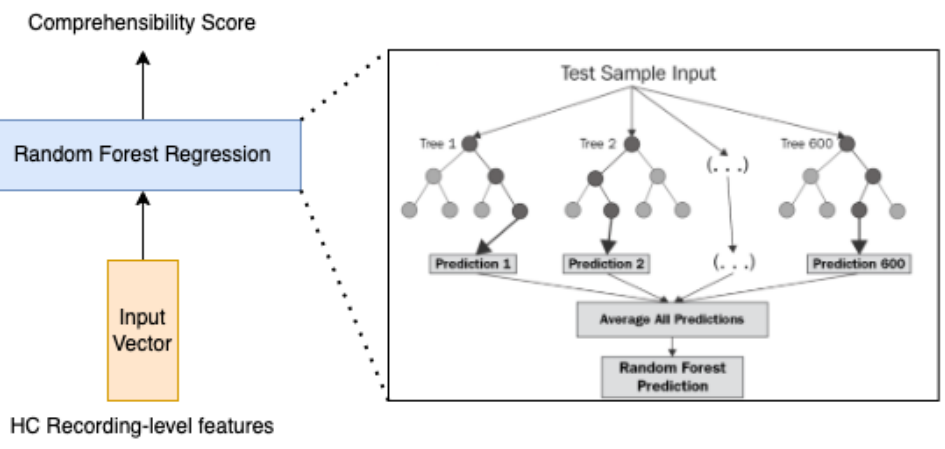
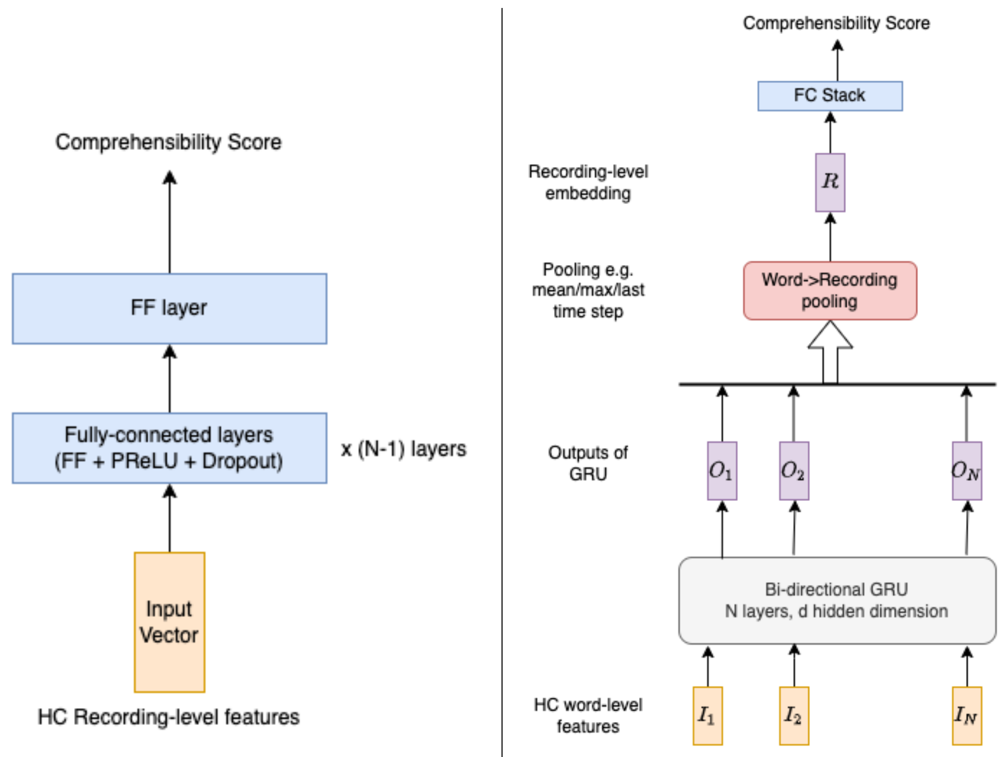


Figure 2.1: An example of a Random Forest Classifier with 600 trees. The predictions of all the trees is averaged to obtain the final score. Figure reproduced from [5].


Comprehensibility Score
FF layer
Fully-connected layers (FF + PReLU + Dropout)
x (N-1) layers
Input Vector
HC Recording-level features
Comprehensibility Score
FC Stack
Recording-level embedding
R
Pooling e.g. mean/max/last time step
Word->Recording pooling
Outputs of GRU
O_1
O_2
O_N
Bi-directional GRU N layers, d hidden dimension
HC word-level features
I_1
I_2
I_N

(a) Recording-level MLP

(b) Word-level BGRU

Figure 2.2: Deep learning alternatives to RFC. Both models accept HC features as input and predict the comprehensibility rating.

2.3.1 Loss functions

We experimented with 3 loss functions: Mean Squared Error (MSE), Pearson loss (= 1 - Pearson correlation) and Concordance Correlation Loss or CCL (= 1 - CCC). Although it sounds counter-intuitive, it has been shown in literature that minimising MSE does not necessarily maximise CCC [28]. Studies such as [29] have demonstrated the effectiveness of using CCL to maximise CCC (instead of using MSE as a loss for maximising CCC). Since we consider CCC to be our main evaluation metric, we use CCL for all experiments.[3]

2.3.2 Hyperparameters and tuning

After trying out various activation functions such as ReLU, PReLU, Swish and Tanh, we stick to PReLU (due to a slight improvement over ReLU) for all experiments. On extensive tuning, we found a dropout probability of 0.2 to be optimal and is hence fixed for all FC stacks.

For training, Adam [30] optimizer is used. The learning rate is set to 0.001 unless specified otherwise. Batch size is set to 128 (after tuning) and models are trained on a single NVIDIA GeForce GTX 1080. Early stopping is used on the validation set with patience set to 10 epochs i.e. if the CCC does not increase by more than 0.005 for 10 epochs, we stop training and choose the model with the best performance observed so far for testing. The random seeds for *PyTorch*, *Numpy* and the default *Random* library in Python are manually set for reproducibility purposes.

All hyperparameter tuning experiments are carried out using the Optuna library[31]. A range for each hyperparameter is chosen based on heuristics e.g. the range of number of FC layers is set to [1, 8] for the recording-level MLP. For learning rate, we consider the range [1e-5, 5e-3]. The library samples randomly from the given range and carries out an experiment. In the initial stages, it explores the range while with more runs, it starts exploiting the best hyperparameters found till then. Best models are chosen on the basis of **validation CCC**.

[3]We experimentally observed that for maximising Pearson correlation, MSE loss gave the best results while Pearson loss led to very poor performance. For maximising CCC, CCL proved to be the best among the three.

Dataset	Features	Model	Val CCC	Test CCC	Test Pearson
ASR	Lexical miscues	RFC		0.651	0.694
		MLP	0.725	0.713	0.723
	A-P contour+Prosodic miscue+Pause	RFC		0.697	0.754
		MLP	0.742	0.72	0.748
	All	RFC		0.736	0.774
		MLP	0.769	0.767	**0.777**
FA	Lexical Miscue	RFC			0.733
		MLP	0.761	0.749	0.762
	A-P contour+Prosodic miscue+Pause	RFC			0.756
		MLP	0.746	0.727	0.76
	All	RFC			0.794
		MLP	0.793	0.789	**0.796**

Table 2.2: Results on replacing the baseline RFC with a MLP consisting of a stack of fully-connected layers operating on recording-level hand-crafted features. Performance on both FA and ASR datasets is reported. Note that validation performance is not reported for the RFC model. Also, Test CCC could not be computed on the RFC models trained on the FA dataset.

2.4 Results

In this section, we report performance of the RFC and deep learning models on various HC feature sets.

RFC is trained using MSE loss, which does not always maximise CCC as discussed previously [28]. Hence, Test Pearson corr. is a more appropriate metric for comparing the performance of a model with that of RFC.

The results of replacing the RFC with a MLP, operating on recording-level HC features, can be found in table 2.2. We find that on simply replacing the RFC with a MLP, there is a noticeable improvement in performance for the *Lexical miscues* feature set (on both FA and ASR datasets). On the other hand, the non-miscue features do not benefit from the deep learning

MLP baseline. Also, on using all feature sets, performance of RFC and MLP is similar. The performance jump on using the FA dataset instead of ASR dataset is noticeable for the lexical miscue feature set. This is expected because as discussed in subsection 1.2.2, the FA dataset is more accurate than ASR for miscue features such as number of miscues, insertions, hesitations, etc. On the other hand, acoustic features are not heavily influenced by the differences in the alignments obtained from FA and ASR dataset. As a result, there is no noticeable difference in the performance of the MLP trained on ASR vs FA dataset. Lastly, for both datasets and for both models (RFC and MLP), combining the two distinct feature sets improves performance. This indicates complementary information among the feature sets.

Dataset	**Word feature set(s)**	**Val CCC**	**Test CCC**	**Test Pearson**
ASR	Acoustic	0.654	0.659	0.676
	LexId	0.394	-	-
	LexMisc	0.341	-	-
	Duration	0.700	0.703	0.714
	PEDMis	0.555	0.554	0.558
	LexId+LexMisc+Duration	0.722	0.727	0.738
	LexId+LexMisc+Duration +Acoustic	0.738	0.741	0.751
	LexId+LexMisc+Duration +PEDMisc	**0.751**	**0.753**	**0.762**
	LexId+LexMisc+Duration Acoustic+PEDMisc	0.747	0.750	0.761
FA	Acoustic	0.648	0.653	0.672
	LexId	0.412	-	-
	LexMisc	0.580	0.581	0.584
	Duration	0.681	0.685	0.697
	PEDMis	0.605	0.603	0.608
	LexId+LexMisc+Duration	0.741	0.744	0.755
	LexId+LexMisc+Duration +Acoustic	0.745	0.75	0.762
	LexId+LexMisc+Duration +PEDMisc	**0.763**	0.766	0.775
	LexId+LexMisc+Duration Acoustic+PEDMisc	0.762	**0.767**	**0.777**

Table 2.3: Results on using various word-level HC feature sets. - indicates unstable training in which case we do not report test performance since models fail to converge.

On going from from recording-level to word-level features, we see a slight drop in performance, as seen in table 2.3. Given below are some key insights obtained on examining the performance of individual feature sets:

- Acoustic and duration features, on their own, give good performance on both FA and ASR datasets. On adding lexical identity and lexical miscue features, we see a further improvement in performance for both datasets.
- A RNN trained only on lexical identity features fails to converge because word identity features such as POS tags do not encode any information about the fluency of the utterance.
- Performance on the *A-P contour* subset for both FA and ASR datasets is similar. This is expected since the word-level acoustic aggregates will only slightly vary due to differences in the alignment. This is consistent with the observation regarding performance of MLP on recording-level acoustic HC features in case of ASR and FA datasets.
- Features such as *Lexical miscue* benefit greatly from manual transcription. The proposed RNN could not be trained on ASR-based *Lexical miscue* features while it was possible to do so with the FA version of the *Lexical miscue* feature set. Also, on including feature sets which includes lexical information (such as Lexical miscue and Lexical identity), a model trained on FA dataset outperforms it's counterpart trained on the ASR dataset.
- PED miscue feature set brings additional value since we observe an improvement on concatenating them with other feature sets.
- On using all feature sets, we see a drop of 0.022 when moving from recording-level features to word-level features for the FA dataset; for the ASR version, we see a similar drop of around 0.026. Note that all recording-level feature sets (except a subset of the A-P contour features) are obtained from an intermediate word-level stage (figure 8.1 in [1]). Thus, although the word-level RNN has more capacity to extract important features on its own from the fine-grained word-level features, it has access to slightly lesser information than the MLP. As a result, comparing the final performance of the MLP and the RNN is difficult.[4]

[4]We can further split the recording-level A-P contour feature set into two subsets: those which are aggregated using word-level A-P contour features and those which are obtained directly from frame-level features. Then, training an MLP by leaving out the latter subset will possibly lead to a fairer comparison.

Chapter 3

Wav2vec2.0-based architecture

Large datasets have been a crucial factor for the rise of deep learning. Deep models for tasks such as ASR are trained on very large corpora (e.g. LibriSpeech, a popular benchmark for ASR consists of 960 hours of speech). However, as discussed previously, we only have 10 hours of labelled data. Training models from scratch with such a limited dataset is difficult. To alleviate this issue, we exploit a recently proposed pre-trained model called Wav2vec2.0 [6].

Wav2vec2.0 is a self-supervised model pre-trained on a large speech corpus. The architecture is shown in figure 3.1. It consists of a convolutional neural network (CNN) and a transformer. The CNN has 7 convolutional blocks with a receptive field of 400 samples or 25 ms of audio (16 KHz sampling rate). The raw waveform is encoded by the CNN and a latent speech representation is generated every 20 ms (based on the stride of the CNN kernels). These latent speech representations are then fed to a stack of transformer encoders.

The entire model (CNN and transformer) is trained end-to-end by masking some of the latent representations and asking the transformer to predict their quantised versions from the unmasked representations. This follows the masked language model pre-training paradigm which is popular in NLP and is used for pre-training large language models such as BERT [32] in a self-supervised fashion. Since the pre-training procedure is task-agnostic (not tuned for any particular task such as ASR, emotion recognition, speaker verification, etc.), the model learns a robust representation of speech at the frame-level[1]. One can subsequently use these embeddings for any downstream task. The original Wav2vec2.0 paper [6] reported competitive performance for ASR on LibriSpeech by using just 10 minutes of labelled data. A simple linear projection layer is added on top of the wav2vec embeddings to predict phones and the model is fine-tuned using CTC loss. The simplicity of the projection layer hints towards the information-rich nature of the wav2vec representations.

Apart from ASR, Wav2vec2.0 has also been used for related tasks such as emotion recognition and speaker recognition. In [2, 33], a pre-trained wav2vec model is used to extract frame-level features. They are transformed using fully-connected layers, CNNs or LSTMs and pooled using mechanisms such as mean pooling to obtain a recording-level representation. This representation is passed through a linear layer to obtain the final score. They reported a substantial improvement over acoustic hand-crafted features such as MFCCs and eGeMAPS for the task of speech emotion recognition. In [34], wav2vec is used as the acoustic backbone while camemBERT (an NLP model) is used for analysing the text for predicting emotion labels every 250 ms. Wav2vec outperformed MFCC and eGeMaps by a huge margin on both English and French recordings. Moreover, on fusing scores from both modalities, the authors report a further jump in performance. A bottleneck approach is presented in [22] for disentangling prosodic information from phonemic content and speaker identity for the task of emotion recognition. Wav2vec features are used for extracting the prosodic embedding. For speaker recognition, various pooling and wav2vec fine-tuning methodologies are presented in [35]. Similar to comprehensibility rating, the above tasks rely on prosodic aspects of speech. Hence, the success of wav2vec on these tasks motivated us to explore this model for our task.

3.1 System architectures

In this section, we discuss two architectures which use wav2vec. The first, called W2Vanilla, requires only the waveform as input. The second, termed W2VAligned, introduces an addi-

[1]Unlike traditional features such as MFCCs, which have a limited context of say 25 ms, the transformer's self-attention mechanism has access to the entire utterance. Hence, although the representations are generated at the frame-level, they are influenced by the entire utterance and not just it's neighbourhood.

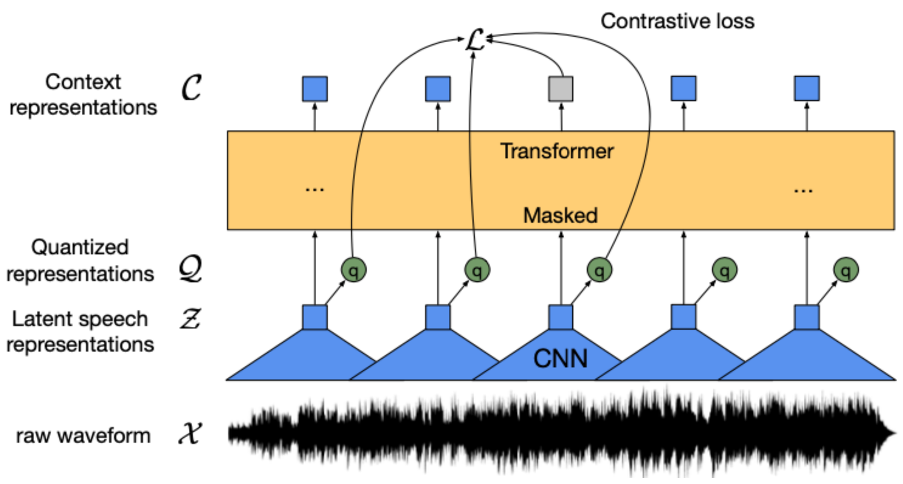


Figure 3.1: The original Wav2vec2.0 architecture which consists of a CNN, a transfomer and a quantisation module. It is trained on a vast corpus of unlabelled speech using contrastive loss. Figure reproduced from [6].

tional pooling stage at the word-level using word alignments, giving us access to word-level representations. Both architectures are discussed next.

3.1.1 W2Vanilla

Our first wav2vec-based architecture (called Vanilla Wav2vec2.0 or W2Vanilla) only requires the waveform from the recording as input. We extract wav2vec representations at the frame-level, pass them through a stack of fully-connected (FC) layers and mean-pool them across the utterance to get a single utterance-level embedding. Each layer in the stack of fully connected layers consists of a feedforward network, an activation function (PReLU in our case) and dropout. The pooled embedding is passed through another stack of FC layers with one output neuron in the final layer, which on Sigmoid activation, can be interpreted as the comprehensibility score. The W2Vanilla architecture is given in figure 3.2. Key hyperparameters of this architecture are the two FC stacks, whose depth and number of hidden units in each hidden layer need to be tuned.

Mean pooling the frame-level representations to obtain a recording-level representation might sound too naive since it leads to loss of temporal sequence information. We tried various deep learning blocks such as GRUs [36] and Transformers [37] on top of the frame-level representations to capture temporal information but it did not lead to any performance improvement. A potential explanation for this is the self-attention mechanism of the Wav2vec2.0 transformer: since it has access to the entire utterance, it can also implicitly represent temporal dependencies in it's frame-level outputs.[2]

3.1.2 W2VAligned

Instead of pooling the frame-level representation across the entire utterance, we introduce an intermediate hierarchical pooling stage at the word level. Word boundaries are obtained by force-aligning the ASR-decoded text or the manually transcribed text. The wav2vec embeddings are then pooled for each word separately to get a word-level representation. For inter-word pauses, we can include either the pause before or after the word in it's pooled representation. We choose to include the pause after the word since it can assist the model in detecting phrase breaks, an important prosodic event[3]. For pooling, we can use parametric models such as RNN, CNN, etc. or non-parametric techniques such as mean pooling, max pooling, etc. The word-level representations are passed through a stack of FC layers before pooling them again using the previously mentioned techniques to get a recording-level representation.

By introducing such pooling, we can now bring in word-level HC features discussed in section 5.1.1. Refer to figure 3.4 for the architecture. Apart from the input FC stack which accepts wav2vec features and the final FC stack which outputs the comprehensibility score (similar to W2Vanilla), we now have an additional FC stack (for transforming the word-level representations) which requires hyperparameter tuning.

[2]The architecture proposed in [38] for speech emotion recognition is almost identical to W2Vanilla. It also uses mean pooling to construct a recording-level representation from frame-level representations. We spoke to the authors and found that they also derived a similar conclusion from their own experiments.

[3]Including both is also possible but since we use mean pooling to aggregate frame-level frames to obtain a word-level representation, the model will not be able to distinguish between pre-word and post-word pauses.

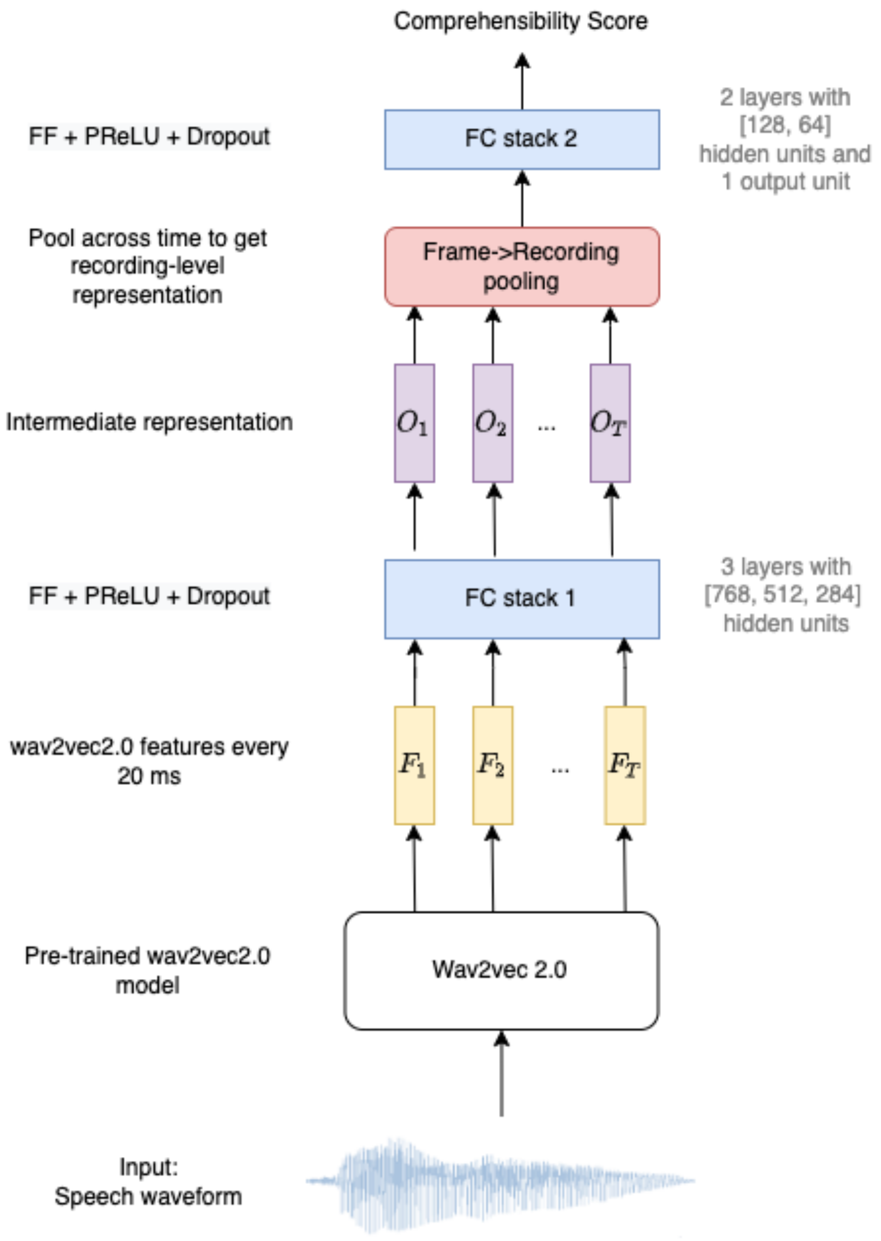

Comprehensibility Score
FF + PReLU + Dropout
FC stack 2
2 layers with [128, 64] hidden units and 1 output unit
Pool across time to get recording-level representation
Frame->Recording pooling
Intermediate representation
O_1
O_2
...
O_T
FF + PReLU + Dropout
FC stack 1
3 layers with [768, 512, 284] hidden units
wav2vec2.0 features every 20 ms
F_1
F_2
...
F_T
Pre-trained wav2vec2.0 model
Wav2vec 2.0
Input: Speech waveform

Figure 3.2: The W2Vanilla architecture consists of a pre-trained wav2vec feature extractor, three stacks of fully-connected layers and a pooling module which computes a recording-level embedding from the frame-level embeddings. The pooling mechanism could be trivial such as mean/max pooling or could be parametric such as a RNN or a CNN. The depth and number of units in both the FC stacks is mentioned next to the stacks. Unless mentioned otherwise, the depth and number of units in the first FC stack (which accepts the wav2vec features) is fixed for all subsequent models.

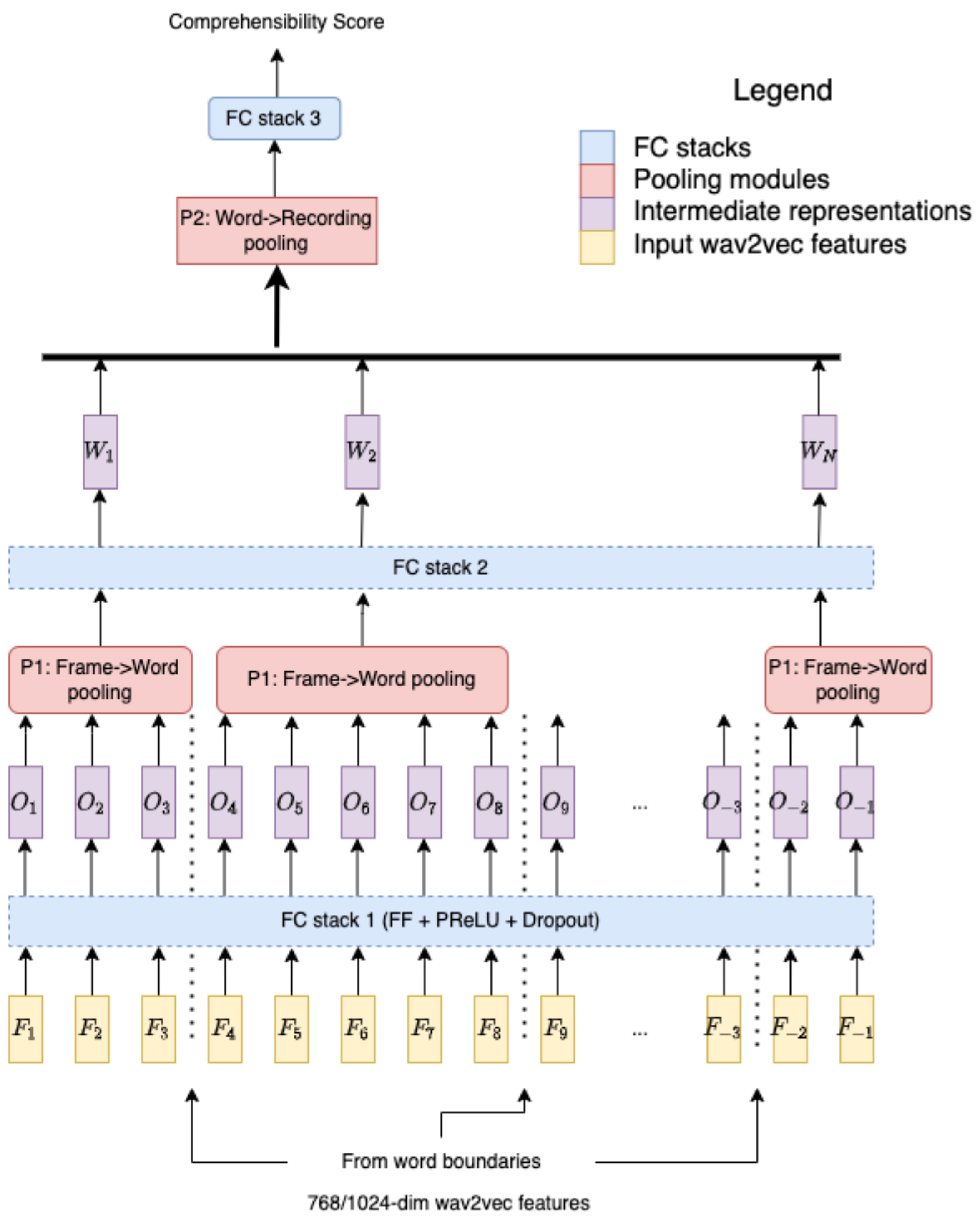

Comprehensibility Score
FC stack 3
P2: Word->Recording pooling
Legend
FC stacks
Pooling modules
Intermediate representations
Input wav2vec features
W_1
W_2
W_N
FC stack 2
P1: Frame->Word pooling
P1: Frame->Word pooling
P1: Frame->Word pooling
O_1 O_2 O_3 O_4 O_5 O_6 O_7 O_8 O_9 ... O_{-3} O_{-2} O_{-1}
FC stack 1 (FF + PReLU + Dropout)
F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9 ... F_{-3} F_{-2} F_{-1}
From word boundaries
768/1024-dim wav2vec features

Figure 3.3: The W2VAligned architecture extends the W2Vanilla architecture with an additional word-level representation. We now have two pooling modules and three FC stacks. The hyper-parameters of FC stacks 1 and 3 (depth and number of hidden units) is the same as W2Vanilla.

3.1.3 Effectiveness of layers

There are two open-source wav2vec pre-trained models available on HuggingFace: *wav2vec2-base*[4] and *wav2vec2-large*[5]. They differ in two aspects: Firstly, the size of the transformer layers: base has a stack of 12 transformer layers while large has 24 layers stacked on top of each other. Secondly, base generates embeddings of dimension 768 while large generates 1024-dimensional embeddings. Both are pre-trained on the same dataset (960 hours of LibriSpeech).

The output of each layer of the transformer encoder is fed as input to the next layer. We can extract frame-level representations from either the last layer (output) or an intermediate layer. Previous works have demonstrated the effectiveness of using intermediate layer representations. For emotion recognition, [2] learned a weighted combination of the transformer layer outputs and discovered that higher weights are assigned to intermediate layers. [39] investigated the performance of Audio ALBERT for speaker verification and phoneme classification as a function of the layer and reported different layers as being optimal for different tasks.

This motivated us to extract representations from various layers of the transformer for our task. Although [2] learnt a weighted combination of the layers, we are unable to do so due to computational constraints. Instead, we try out two additional techniques:

- Mean pooling: at each time-step, take the average of the outputs of each layer of the transformer encoder to obtain the final frame-level embedding.

- Gaussian pooling: inspired by the weights learnt in [2], we use a Gaussian distribution centred at the middle (e.g. between layer 12 and layer 13 for *wav2vec2-large*). The weights are normalised so that they sum to 1. The variance controls how rapidly the weights decay as we move away from the middle layer. An example distribution is given below:

[4] `https://huggingface.co/facebook/wav2vec2-base-960h`
[5] `https://huggingface.co/facebook/wav2vec2-large-960h`

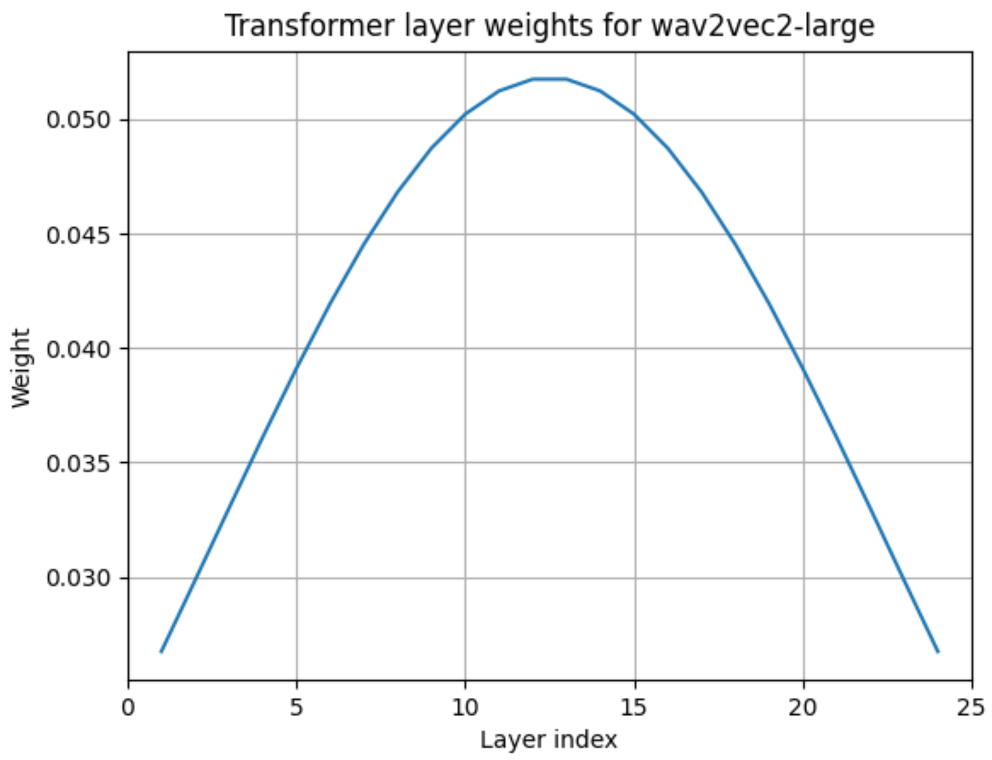


Figure 3.4: Layer-wise weights generated using a Gaussian with standard deviation of 10.

3.1.4 Fine-tuning the transformer

Generally, apart from the linear layers on top of wav2vec, the transformer module is also fine-tuned for the downstream task to increase performance [38]. However, due to computational constraints, we freeze the transformer and tune only the FC stacks. We did try tuning the transformer layers by reducing the batch size and clipping the audios to only 10 seconds so that the data could be loaded on our GPU. However, the performance was poor and it took more than a day to train the entire model. We believe that fine-tuning the transformer layers will require extensive experimentation with hyperparameters such as learning rate, learning rate schedulers, etc.

3.2 Experimental results

In this section, we discuss the performance of W2Vanilla, W2VAligned and the layer-wise performance of *wav2vec2-large*.

3.2.1 W2Vanilla

Which layer	Val CCC	Test CCC	Test Pearson
Output	0.765	0.759	0.772
Mean	0.800	0.786	0.808
Gaussian	**0.801**	**0.807**	**0.813**

Table 3.1: The performance of *wav2vec2-base* on using various transformer outputs. Taking the mean of the embeddings across the layers or a Gaussian weighted embedding performs significantly better than the output of the transformer. This is in line with the observation in [2] that for non-ASR tasks, intermediate layer representation encode crucial prosodic information.

In table 3.1, we report the performance of W2Vanilla operating on *wav2vec2-base* embeddings. Mean pooling and Gaussian pooling outperform the RFC and MLP, which are trained on both acoustic and lexical feature sets. The implications of this result cannot be overstated; the wav2vec-based model, without explicitly incorporating any knowledge of the canonical text and the uttered lexical content, already outperforms extensive hand-engineered features which incorporate both lexical and acoustic information. A potential explanation is that wav2vec, pre-trained on a vast corpus, also captures some degree of lexical content, pause features, duration features and prosodic events. Refer to section 4.4.1 for a more extensive discussion on the nature of information which is captured by wav2vec.

Next, we replace the base model (12 layers, 768-dimensional embeddings) with the large model (24 layers, 1024-dimensional embeddings). Along with mean pooling and Gaussian pooling, we perform a layer-wise analysis of the performance of *wav2vec2-large* and the results are presented in table 3.2. Firstly, we see an improvement in performance on using both

Which layer	Val CCC	Test CCC	Test Pearson
1	0.723	0.696	0.721
5	0.732	0.558	0.644
10	0.803	0.803	0.813
15	0.813	0.803	0.816
16	0.814	0.809	0.817
17	**0.816**	0.808	0.821
18	0.809	0.808	0.817
21	0.787	0.787	0.796
Output	0.77	0.772	0.778
Mean	0.813	0.814	**0.823**
Gaussian (20)	0.813	**0.817**	0.822

Table 3.2: The performance of *wav2vec2-large* on using various transformer representations.

mean and Gaussian pooling as we move from *wav2vec2-base* to *wav2vec2-large*. This can be attributed to the larger capacity of the *wav2vec2-large* network.

The layer-wise analysis reveals another interesting property: we find a sweet spot at layer 17; performance for comprehensibility prediction drops on using the transformer outputs of layers before or after layer 17[6]. The performance of the output layer (layer 24) is noticeably lower than layer 17, thereby hinting towards differences in the nature of information which is being stored in various layers of the wav2vec transformer.

3.2.2 W2VAligned

In this section, we report the performance of the W2VAligned architecture, in which an additional pooling stage from the frame-level representations to word-level representations is introduced. As of now, we use mean pooling to pool the frame-level representations and obtain a word-level representations. Refer to Appendix B for more details regarding this choice.

[6]Although layer 17 gives the best validation performance, we use mean pooling for all subsequent experiments since layer-wise analysis and Gaussian pooling was carried out later on.

Alignment	Val CCC	Test CCC	Test Pearson
W2Vanilla	0.813	0.814	0.823
FA	0.808	0.778	0.814
ASR	0.804	0.781	0.815

Table 3.3: Results of W2VAligned on using alignments from either the manually transcribed text or the ASR decoded text.

Counter-intuitively, performance slightly reduces on introducing this intermediate pooling stage. A possible explanation is the rigid pooling mechanism, coupled with erroneous alignments, leads to erroneous pooling of frame-level representations to obtain word-level representations. Moreover, validation CCC on using FA dataset is slightly higher than for the manually transcribed dataset while the trend reverses for test CCC. This possibly indicates overfitting caused by precise word alignments of the FA dataset.

This page was intentionally left blank.

Chapter 4

Probing Wav2vec2.0

Hand-crafted features are interpretable by definition. On the other hand, despite outperforming traditional signal processing techniques on a wide range of tasks, deep learning models are opaque. The frame-level embeddings which are extracted from waveform by wav2vec are not directly interpretable. Without understanding the nature of information which is being captured by wav2vec, it is difficult to further push the performance of model by trying to incorporate possibly complementary information.

In [19], known acoustic features such as voicing probability and loudness are regressed from representations of a CNN which is trained for prosodic event detection. To crack open the black box that is wav2vec, we propose a similar approach by adding a stack of linear layers on top of our current architecture. These layers are trained to predict HC features (discussed previously) at various hierarchies.

The motivation for such probing is two-fold:

- By analysing the ability of wav2vec to predict HC features, we can draw conclusions regarding the nature of information which is being encoded by the wav2vec pre-trained model.

- If wav2vec is unable to capture a particular HC feature which is known to be crucial for the task of comprehensibility prediction, we can concatenate such a feature to increase performance.

In the subsequent sections, we discuss models for probing the presence of HC features at various hierarchies.

4.1 Architecture

Frame-level probing can be realised by adding linear layers for each frame-level feature on top of the transformed wav2vec features i.e. output of the first input FC stack. Refer to figure 4.2 (a) for the overall architecture. We use W2Vanilla (proposed in subsection 3.1.1) since word alignments are not required for frame-level probing.

To analyse recording-level information, we again use W2Vanilla and add a separate stack of fully-connected layers for each HC feature as shown in figure 4.2 (b).

For word-level probing, we use the W2VAligned model to extract word-level representations. The structure of the probes is similar: a stack of linear layers is added for each HC feature. The architecture shown in figure 4.1.

4.2 Feature sets

Since wav2vec does not have access to ASR outputs and canonical text, we speculate that it may be difficult to extract features apart from those in the *A-P contours* feature set. However, wav2vec embeddings also capture lexical content because it can perform ASR with very minimal labelled data. Hence, it might not be completely unreasonable to expect wav2vec to predict lexical features such as POS tags. Moreover, although timing information is not explicitly incorporated into our architecture, it is interesting to analyse if wav2vec can capture rate features.

We probe the model for the presence of word-level and frame-level HC features discussed in table 2.1. To narrow down the number of recording-level features, we instead choose a

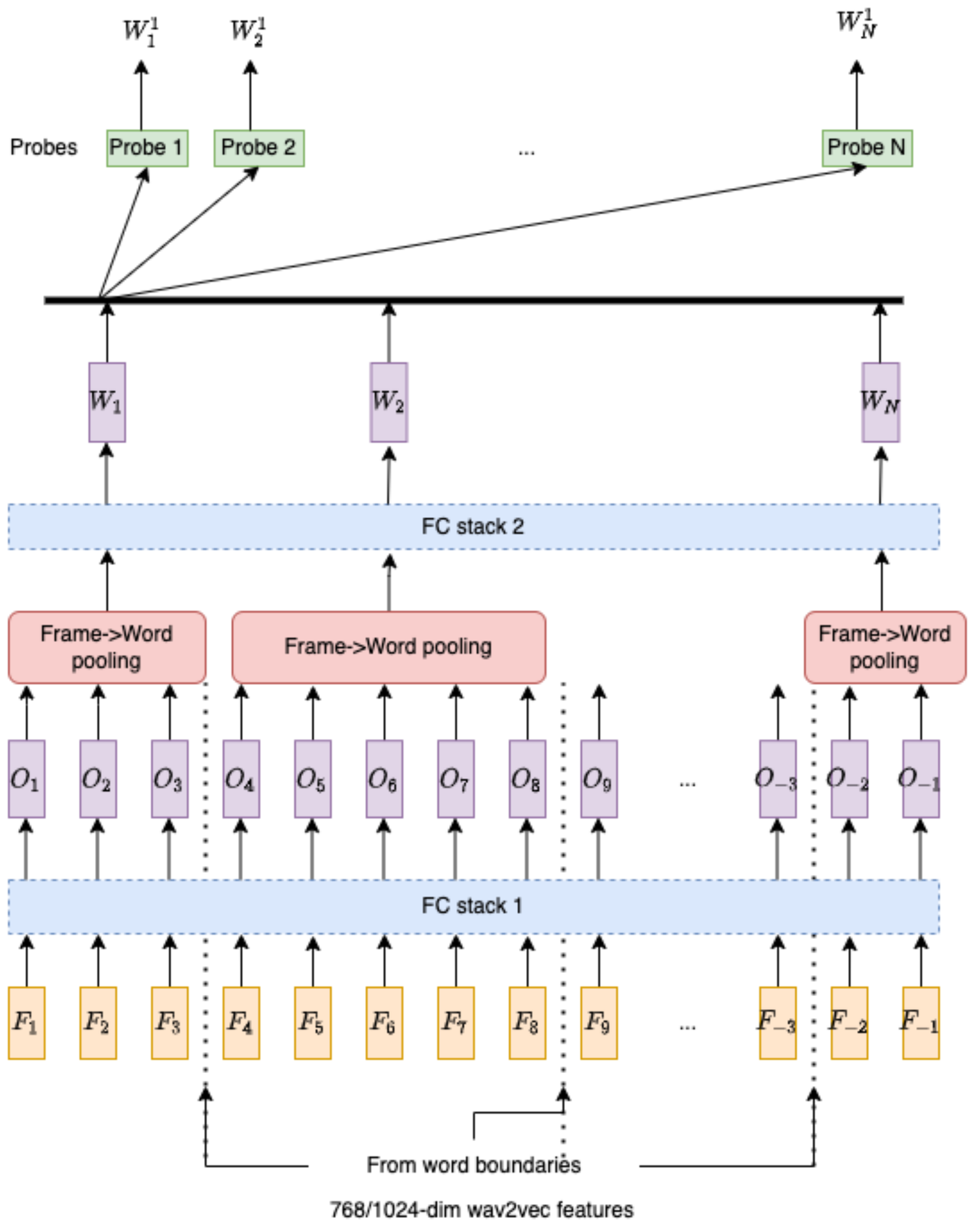
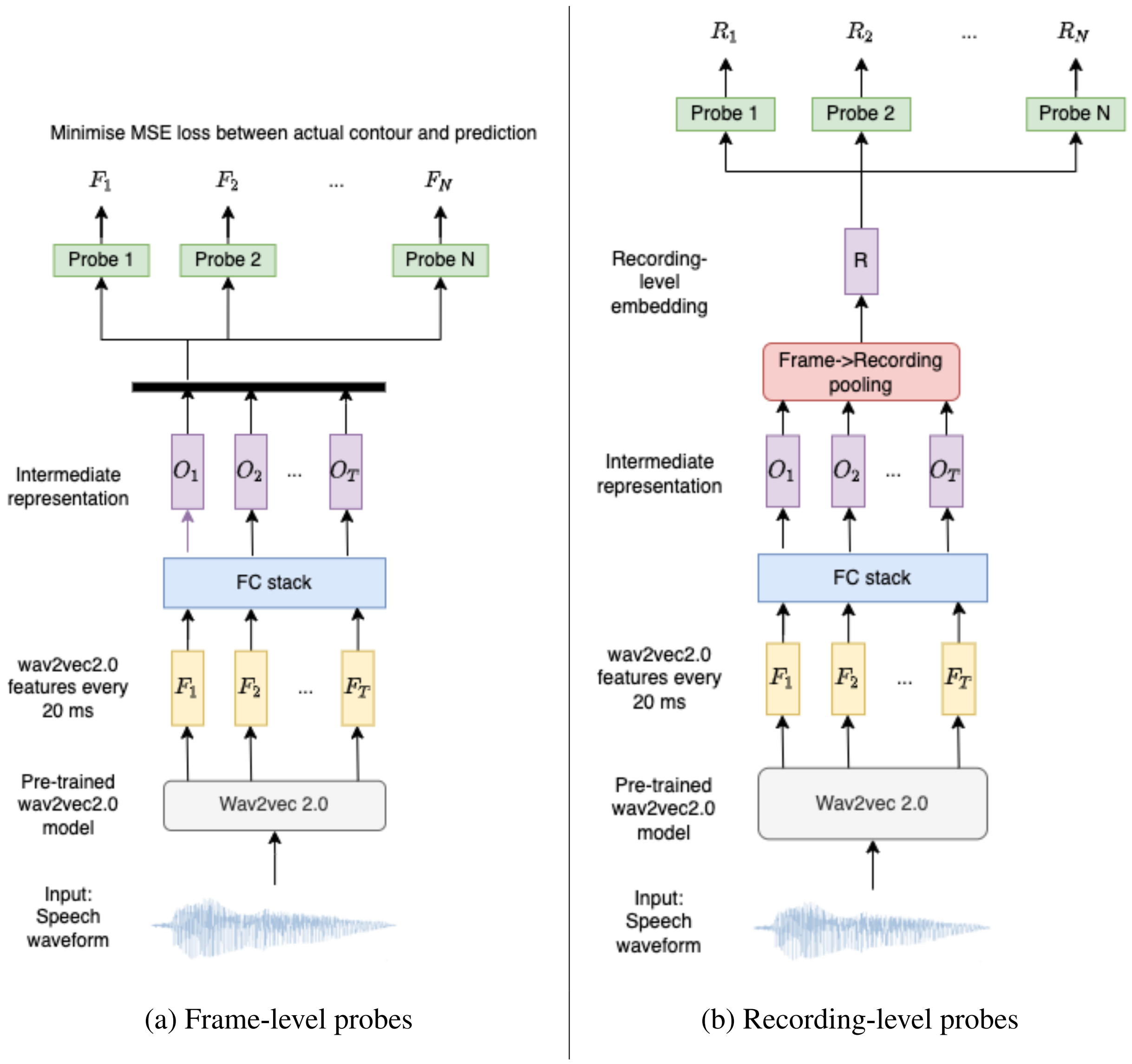

W_1^1
W_2^1
W_N^1
Probes
Probe 1
Probe 2
...
Probe N
W_1
W_2
W_N
FC stack 2
Frame->Word pooling
Frame->Word pooling
Frame->Word pooling
O_1 O_2 O_3 O_4 O_5 O_6 O_7 O_8 O_9 ... O_{-3} O_{-2} O_{-1}
FC stack 1
F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9 ... F_{-3} F_{-2} F_{-1}
From word boundaries
768/1024-dim wav2vec features

Figure 4.1: Architecture for the word-level probing architecture. We use W2Aligned as the backbone since we require word level representations.



(a) Frame-level probes

(b) Recording-level probes

Figure 4.2: Architecture for the recording-level and frame-level probes. Note that we use the W2Vanilla backbone since word alignments are not required for the frame-level and recording-level probing.

compact set of 26/14 features for ASR/FA dataset obtained after RFECV. This set contains features from all feature sets (accuracy, acoustic, prosodic and rate) and is mentioned in table 8.9 of [1].

4.3 Procedure

In this section, we describe the loss function used for training the probes and an important discussion on the implications of training the network from scratch.

4.3.1 Loss function

We minimise the MSE loss between the model prediction and the HC features.

For recording-level features, we average the loss across all the HC features:[1]

$$L_{recording} = \frac{1}{N_R} \sum_{j=1}^{N_R} (R_j - YR_j)^2 \tag{4.1}$$

where N_R is the number of recording-level HC features, R_j is the model's prediction for feature j and YR_j is the feature value (ground truth).

For word-level features, the MSE loss is computed by averaging it across both words and features across the entire utterance:

$$L_{word} = \frac{1}{TN_W} \sum_{i=1}^{T} \sum_{j=1}^{N_W} (W_j^i - YW_j^i)^2 \tag{4.2}$$

where T refers to the number of words in the utterance, N_W is the number of word-level HC features, W_j^i is the model's prediction for feature j of word i and YW_j^i is the ground truth

[1]On tuning only the probes, minimising averaging MSE is equivalent to independently minimising the MSE for each feature since parameters which are shared across the probes are frozen.

feature value.

Similarly, for frame-level features, the MSE loss is computed by averaging it across both frames and features across the entire utterance:

$$L_{frame} = \frac{1}{FN_F}\sum_{i=1}^{F}\sum_{j=1}^{N_F}(F_j^i - YF_j^i)^2 \tag{4.3}$$

where F refers to the total number of frames in the utterance, N_F is the number of frame-level HC features, F_j^i is the model's prediction for feature j of frame i and YF_j^i is the feature value (ground truth).

4.3.2 Training all layers

To analyse the nature of representations learnt by the comprehensibility prediction network, we tune only the prediction heads while keeping the rest of the network fixed. In other words, the blue layers in figure 4.1 and figure 4.2 are initialised with weights from a model trained for comprehensibility prediction on the training data and are then frozen while the ones in green are tuned for minimising the MSE loss between model predictions and HC features.

However, on training the **entire network** from scratch, we can deduce the HC features which can be **potentially** extracted from wav2vec. No comprehensibility labels are used in any stage of the training; only the HC features are used as targets to train the model. Since all parameters of the model can be updated, the performance of this network can be considered to be an approximate upper bound[2] on the ability of our current architecture to extract HC features.

[2]Since the input FC layers (layers in blue in figure 4.2 and 4.1) are shared across features, they need to be robust enough to capture all HC feature targets; hence the term approximate. We can obtain a true upper bound on training an entire model to predict only one feature at a time, assuming there is no overfitting.

4.4 Results

In this section, we discuss the results of the probing experiments. The Pearson correlation of the probe output and the corresponding ground truth HC feature value is computed. We use Pearson instead of MSE/RMSE since the former is invariant to bias and scale of the features, as a result of which we can compare performance across features. The mean Pearson across the 6 folds, along with an error bar indicating standard deviation, is plotted for each feature.

At the word-level, we split the set of feature sets into acoustic (A-P contour) and non-acoustic (Duration, LexId, LexMisc and PEDMisc). The reason for such a split is that we intuitively expect wav2vec to capture the acoustic information while features based on the canonical text and uttered lexical content might be difficult to capture for the purely acoustic wav2vec model.

For each experiment, we report results for two cases discussed in section 4.3.2:

1. Initialising the network with weights trained for comprehensibility and tuning only the linear probes for predicting HC features (yellow contour in all the plots). This is the standard methodology which has been used in previous works such as [19].

2. Tuning the entire network from scratch for predicting the HC features (blue contour in all the plots).

For experiments involving the tuning of only probes (case 1 above), the probes are simple FC stacks with one hidden layer of 128 units. They are intentionally kept shallow so as to ensure that no further feature extraction is performed by the probes [40]. For experiments involving the tuning of the entire network from scratch (case 2 above), the probes are FC stacks with three hidden layers, consisting of [256, 128, 64] units. They are deep so that the model has more freedom to extract as much information as possible in order to predict an upper bound on the ability to extract a given feature.

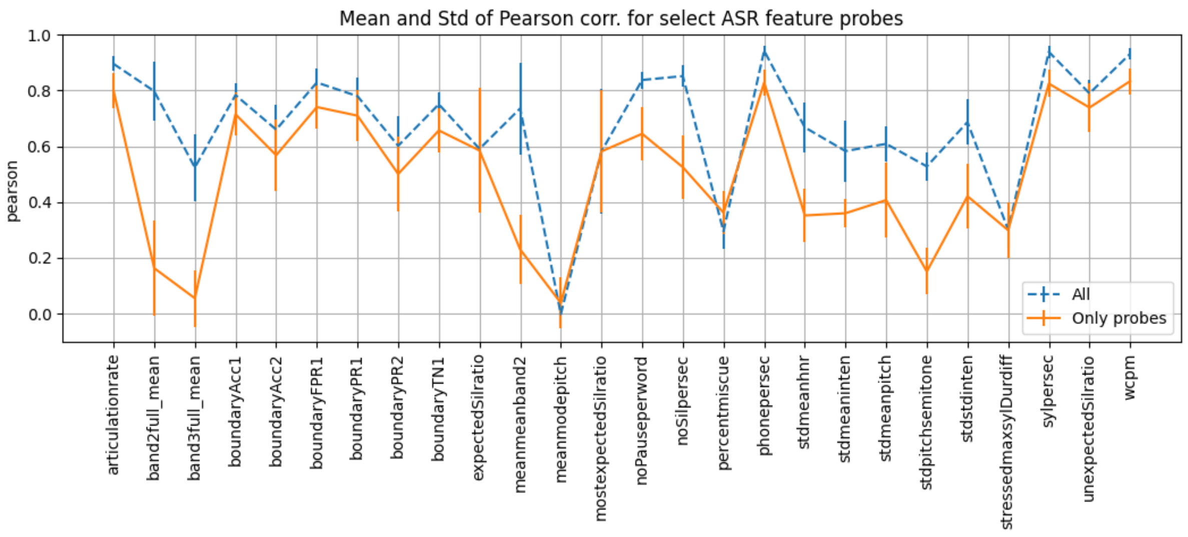


Figure 4.3: Probe results for a compact set of 26 ASR features derived after RFECV. Refer to section 2.1 for more details.

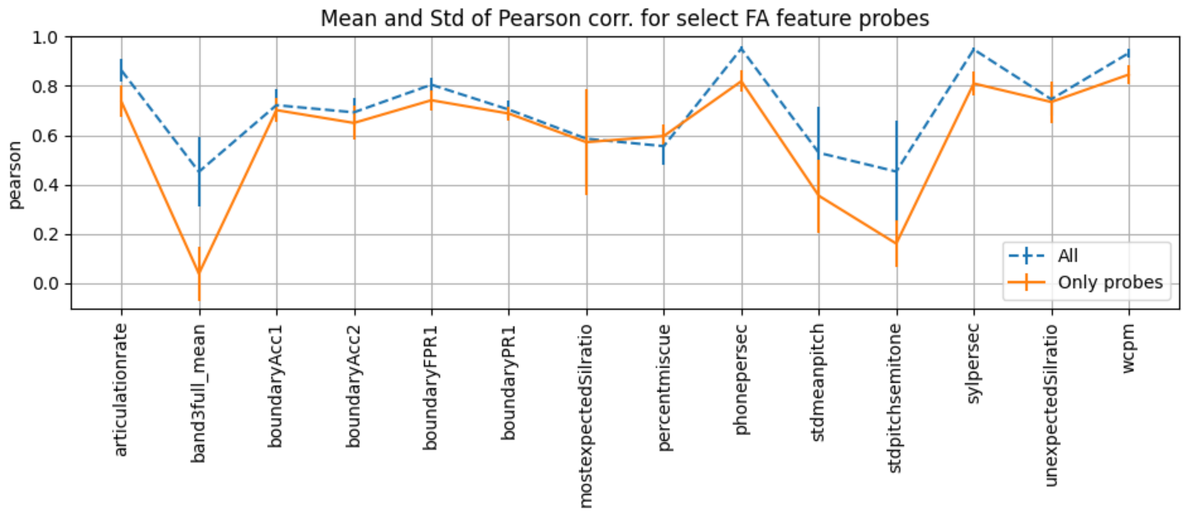


Figure 4.4: Probe results for 14 recording-level features derived from the acoustics and manually transcribed dataset. For more details, please check section 2.1. The number of features differs from the corresponding ASR set due to RFECV.

4.4.1 Recording-level probes

From figures 4.3 and 4.4, we observe that the mean band intensity of specific bands across the entire utterance[3] (*band2full_mean* and *band3full_mean*), mean of word-level statistical mode of F0 (*meanmodepitch*) and recording-level s.d. of F0 semitone (*stdpitchsemitone*) are relatively difficult to extract. The latter two can be explained by the simplistic mean pooling mechanism; any temporal patterns in the low-level features are lost on average pooling the representations. Although wav2vec has the potential to extract band intensities (high correlation in the blue plot for *band2full_mean* and *band3full_mean*), poor performance for the same in the yellow plot implies that the wav2vec model tuned for comprehensibility does not consider band intensities as an important factor in predicting comprehensibility. Percent lexical miscues (*percentmiscue*) is also difficult to extract from wav2vec, which is expected because it has no knowledge of canonical text.

Rate-based features such as WCPM (*wcpm*), syllables per second (*sylpersec*) and phrase boundary aggregates such as accuracy and precision/recall (*boundaryAcc1, boundaryPR1, ...*) are relatively easy to extract. This implies that wav2vec can also capture speech rate.

Similar conclusions can be drawn for the selected FA features when it comes to band intensity, speech rate and prosodic aggregate features. There is one interesting difference though: the performance of the model on lexical miscue percentage (*percentmiscue*) is much higher for the FA dataset (0.6 for FA vs 0.4 for ASR dataset). This is expected since the ground truth feature value for FA based miscues is more accurate than the ASR value. It also highlights the importance of using accurate ground truth targets for probes; inaccurate ground truths will lead to inaccurate conclusions!

4.4.2 Word-level probes

On inspecting the ability of wav2vec to predict word-level acoustic features after training the network from scratch (blue plot), we discover that mean and standard deviations of pitch, intensity, HNR and various bands across a word can be extracted from wav2vec. This indicates that

[3]No word alignments are used; we simply average out the frame-level intensities across the entire utterance.

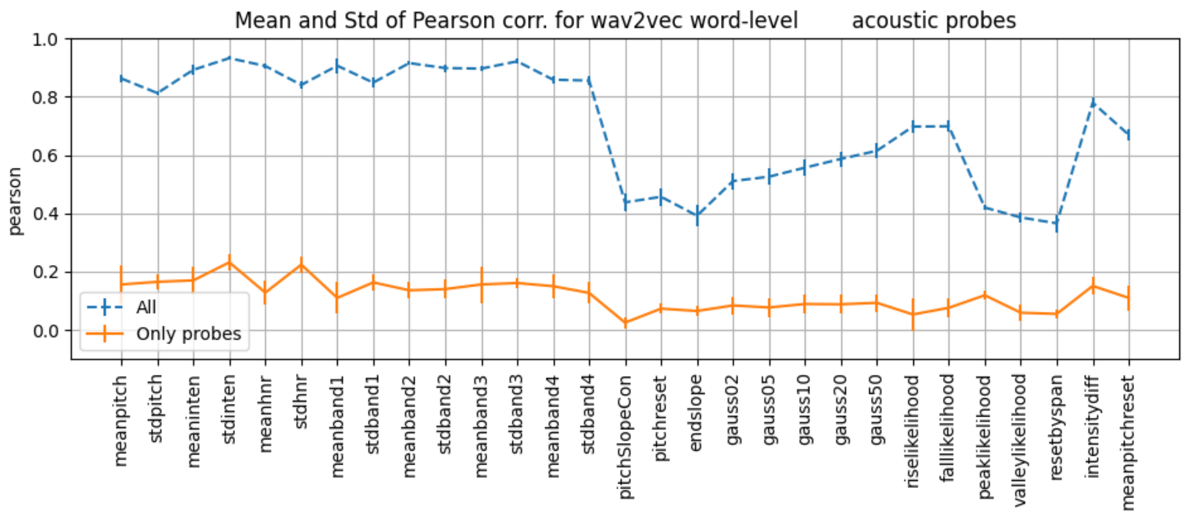


Figure 4.5: Pearson correlation of word-level acoustic HC features and outputs of the probes.

the transformer can learn word boundaries using the self-attention mechanism. On the other hand, pitch contour functionals such as slope, correlation with gaussians of varying standard deviations and peak/valley likelihoods are difficult to extract because our current pooling module consists of simple mean pooling. As a result, temporal pitch information across the word is lost. The smooth increase in correlation as we go from correlation of pitch contour with a sharp Gaussian (*gauss02*) to a flatter Gaussian (*gauss50*) validates this hypothesis since mean pooling is equivalent to correlation with a Gaussian of infinite standard deviation.

We observe a very steep drop in correlation for all HC features (the yellow contour) on freezing the input FC stack and tuning only the probes. This indicates the information encoded in wav2vec features tuned for the task of comprehensibility tell a different story; none of the HC word-level acoustic features are being used by wav2vec for comprehensibility prediction. The large gap between the blue and yellow contour indicates that if such HC features are indeed helpful, incorporating them in an MTL framework should improve performance.

From the word-level non-acoustic probes trained **from scratch** (figure 4.6), we observe that the functionals of silence and syllable durations can be extracted from wav2vec. It points to the ability of the transformer to detect fine-grained syllable boundaries. Also, pauses should be easy to predict from the acoustics.

At first glance, it may seem counter-intuitive that the model does a surprisingly decent job at extracting POS tags (blue plot for features *AU, CC, DT, ..., VB* in figure 4.6). However,

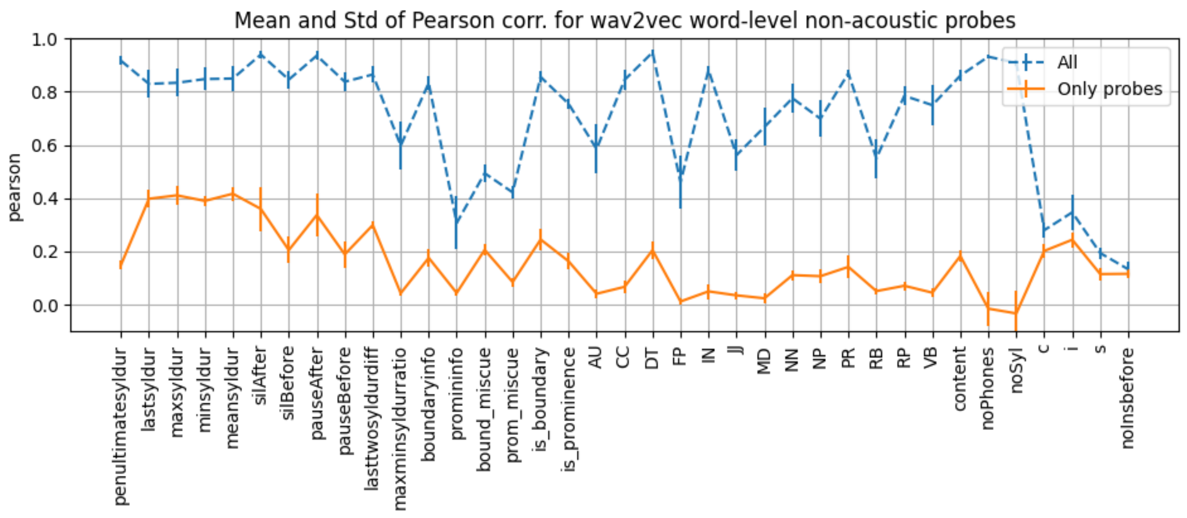


Figure 4.6: Pearson correlation of word-level non-acoustic HC features and outputs of the probes.

wav2vec, with just 10 minutes of labelled data, is shown to perform well for ASR. Therefore, it can exploit some residual linguistic content in the embeddings to predict the POS tags.

The results for the set of 6 prosodic features are quite interesting (the features *boundary-info, prominfo, boundmiscue, prommiscue, is_boundary, is_prominence* in figure 4.6.). Firstly, the model is able to predict the information structure for phrase boundary while there is a drop in the ability to predict the prominence information structure. This is expected because phrasing can be accurately determined from the text while prominence is much more subjective. In fact, knowledge of POS tags can serve as a reliable indicator for phrasing and since the model is able to extract atleast some POS tags quite well, high performance for phrasing information structure is not surprising. The performance for prominence (*is_prominence*) and boundary (*is_boundary*) is also quite high, indicating wav2vec's ability to detect prosodic events.

Lastly, the model fails to predict miscue tags such as correct, inserted and substituted. This is not surprising since it has no knowledge of the canonical text.

4.4.3 Frame-level probes

The results for frame-level probing are depicted in figure 4.7. Both plots indicate that as compared to other frame-level contours, pitch, energy and Sonorant energy are difficult to extract

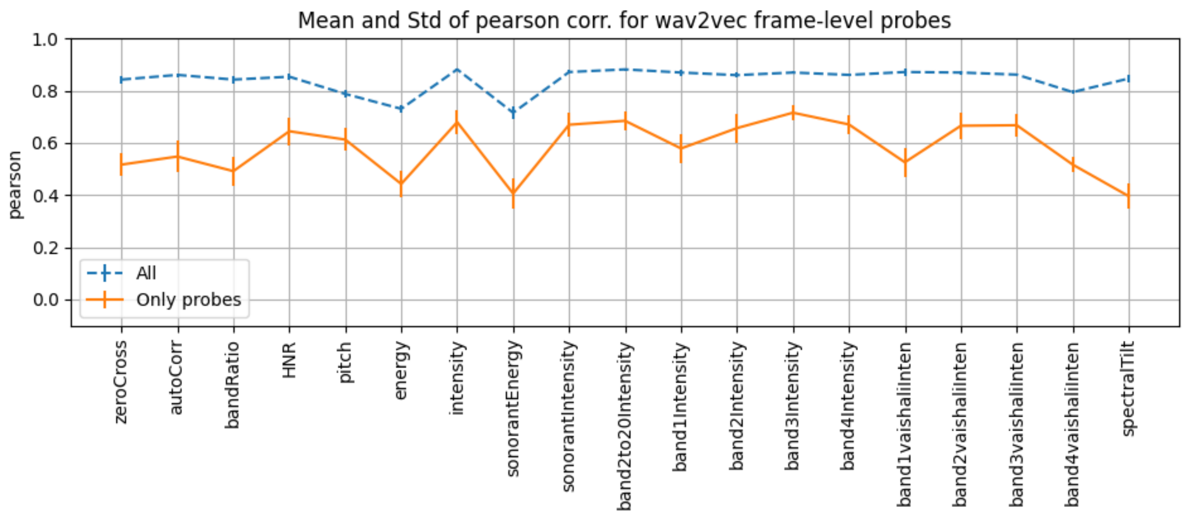


Figure 4.7: Pearson correlation between frame-level contours and probe predictions.

from wav2vec. Poor performance for pitch can be explained by the fact that computation of pitch is not trivial and hence the ground truth itself might be noisy. On the other hand, high correlation for intensity but low correlation for energy is counter-intuitive since intensity is simply the logarithmic energy, normalised by silence energy. Similarly, unlike sonorant intensity, sonorant energy is also difficult to extract. A possible explanation is that the transformer uses self-attention to normalise the utterance with respect to silences.

Except the expected drop in performance, there is no noticeable change in the trend as we move from the blue contour (which represents information which can be potentially extracted from wav2vec on training it from scratch) to the yellow contour (wav2vec tuned for the task of comprehensibility).

4.5 Concluding remarks

Correlation does not imply causation is an important maxim, which is all the more applicable for black-box deep learning models. Current architectures (including the ones proposed in this work) do not explicitly incorporate causality. As a result, correlation with a particular handcrafted feature does not imply that it was responsible for the final prediction. Hence, through the probing procedure explained above, it is erroneous to claim that a particular HC feature is responsible for the prediction, just because we can extract it from tuned wav2vec. Such simplistic probing can only throw light on the nature of information which can be potentially extracted from wav2vec. If a particular HC feature, which has been proven to be crucial for comprehensbility prediction in literature, cannot be extracted after extensive probing, we can think of ways to incorporate it from an external source. This is the key usage of the probing technique discussed above.

This page was intentionally left blank.

Chapter 5

HC feature concatenation

Although one of the key goals of this work is to do away with the extensive hand-engineering involved in extracting features, it is interesting to analyse if they can supply information which is complementary to wav2vec. To do so, we propose two architectures in this chapter.

5.1 Architectures

The first architecture, termed W2VCat (Wav2vec + con**cat**enation), involves the concatenation of hand-crafted features at various hirearchies. The second one, termed RNN Fusion, combines information from wav2vec and a RNN operating on word-level features.

5.1.1 W2VCat

In this subsection, we augment the W2Vanilla model (figure 3.2) with hand-crafted features. By concatenating various HC feature sets derived in the previous work [1] at various hierarchies, we can check for complementary information between wav2vec embeddings and the HC features. The architecture is given in diagram 5.1. HC features at each hierarchy are passed through a separate stack of FC layers before concatenation to extract meaningful task-specific representations.

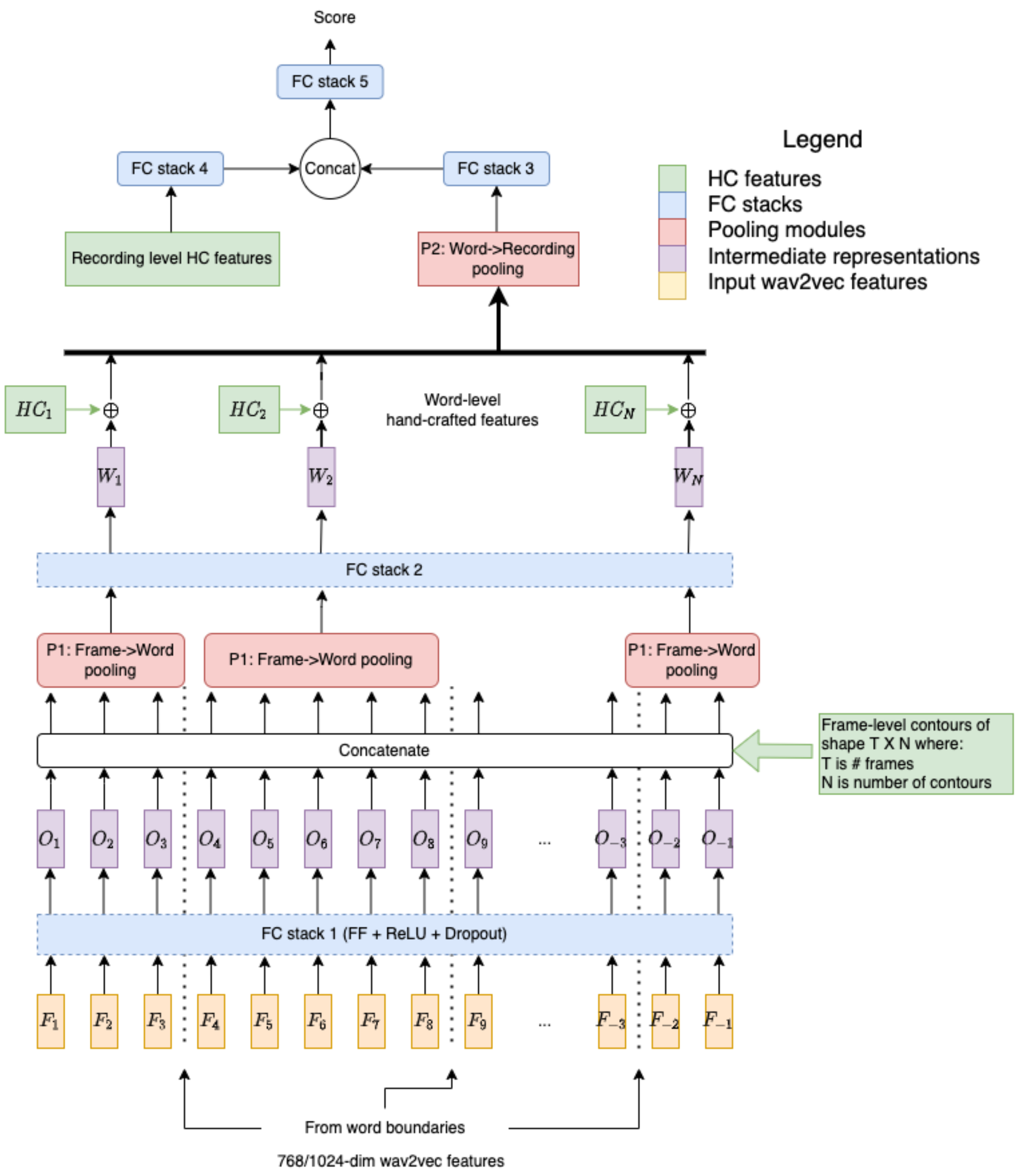
5.1.2 RNN Fusion

Instead of concatenating word-level HC features with the wav2vec word-level representations, we experiment with a parallel RNN branch which operates on the HC features. We use the same RNN which was discussed in section 2.2.2. We fuse information from the two modalities at two different hierarchies: decision-level and embedding-level.

In decision-level fusion, each branch (wav2vec branch and the RNN branch) predicts a comprehensibility score. The two scores are subsequently averaged to get the final prediction, as depicted in figure 5.2. This is similar to the approach proposed in [22], except we do not learn a weighted combination of the scores of the two branches and stick to simple averaging[1]. In embedding-level fusion (figure 5.3), we concatenate the final representation from both modalities and then pass this vector through a stack of FC layers to predict the final comprehensibility rating. This increased flexibility the model to capture relationships between acoustic and lexical modaliities before predicting a score.

Ideally, training the two branches from scratch should maximise performance as each branch can extract complementary information from the input. However, in practice, initialising the weights of the model with a pre-trained network can not only speed up training but also improve performance. Hence, we report performance of the fusion architectures by training each branch separately for the task of comprehensibility prediction and then subsequently fine-tuning both the branches together.

[1]Instead of learning fixed weights for each branch, it would be interesting to explore an attention-based weighting mechanism which generates different weights for each branch depending on the input.



Score
FC stack 5
FC stack 4
Concat
FC stack 3
Recording level HC features
P2: Word->Recording pooling
Legend
HC features
FC stacks
Pooling modules
Intermediate representations
Input wav2vec features
HC_1
HC_2
Word-level hand-crafted features
HC_N
W_1
W_2
W_N
FC stack 2
P1: Frame->Word pooling
P1: Frame->Word pooling
P1: Frame->Word pooling
Concatenate
Frame-level contours of shape T X N where:
T is # frames
N is number of contours
O_1 O_2 O_3 O_4 O_5 O_6 O_7 O_8 O_9 ... O_{-3} O_{-2} O_{-1}
FC stack 1 (FF + ReLU + Dropout)
F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9 ... F_{-3} F_{-2} F_{-1}
From word boundaries
768/1024-dim wav2vec features

Figure 5.1: W2VCat includes the ability to concatenate HC features at various hierarchies. FC stack 1 has the same hyperparameter settings as W2Vanilla and W2VAligned.

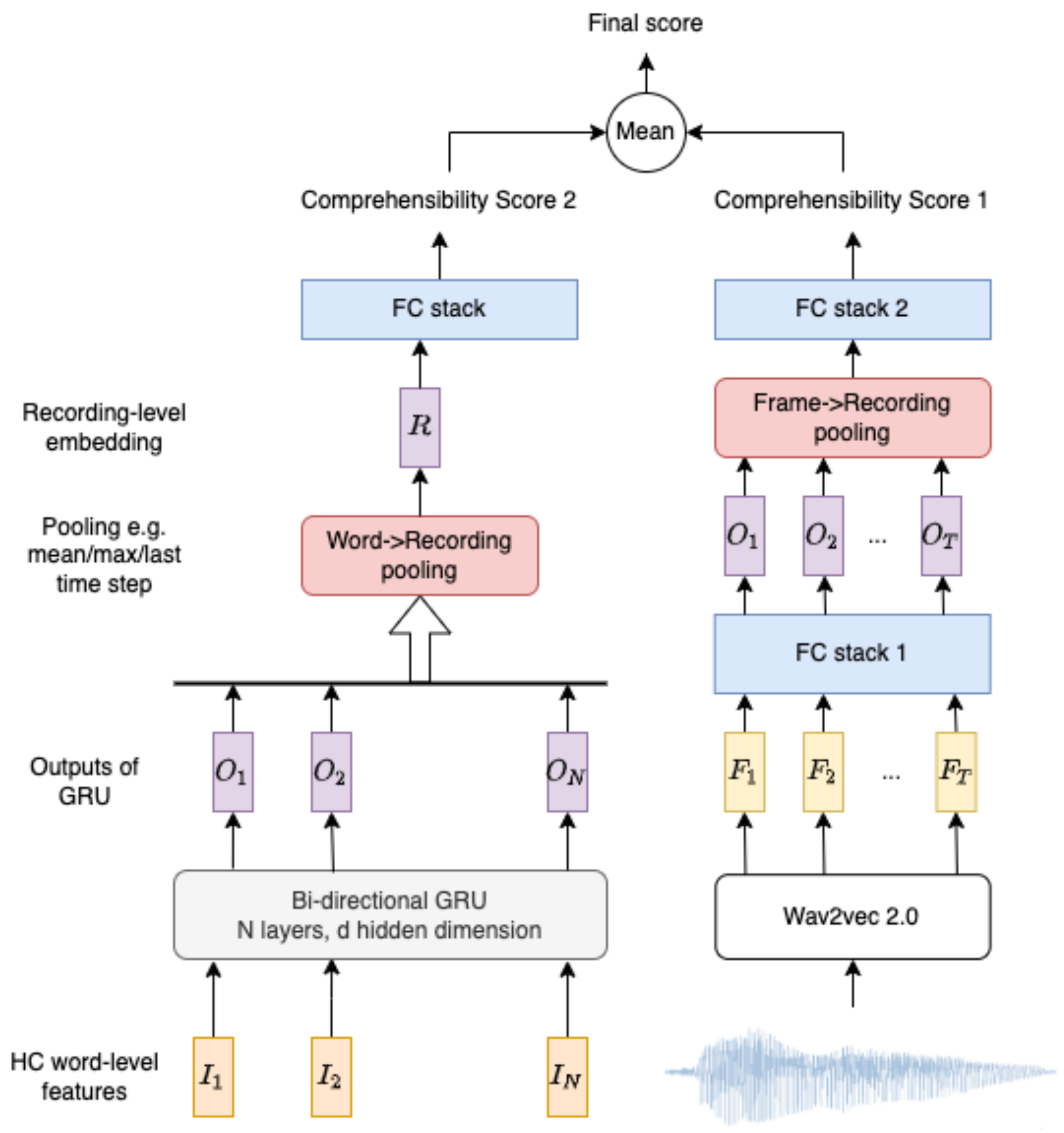


Figure 5.2: In decision-level fusion, the scores from the two branches are averaged to get the final prediction. Both branches are simultaneously trained end-to-end so that each branch can extract complementary information.

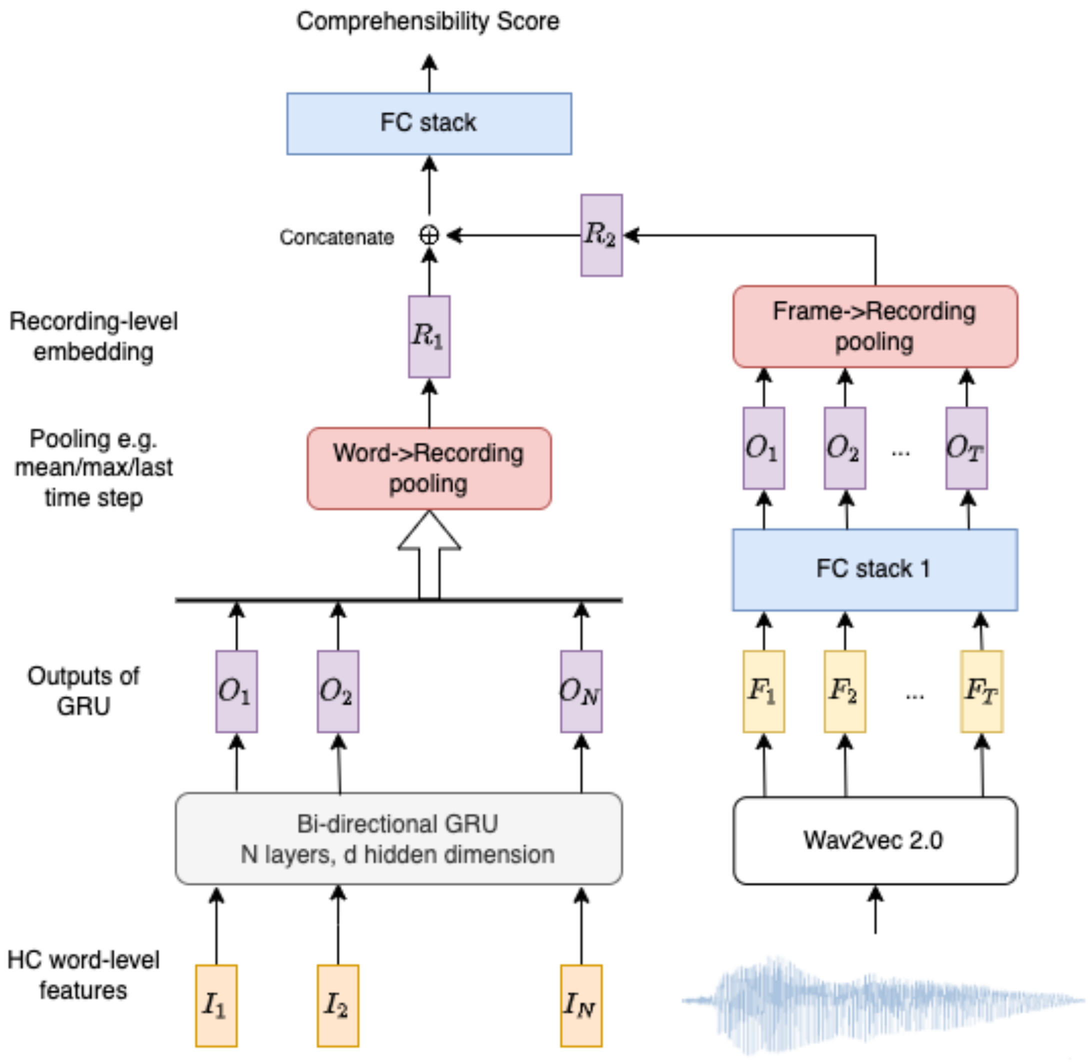


Figure 5.3: In embedding-level fusion, the information from the two branches is fused at a lower-level than decision fusion. The representations from the two branches are concatenated to obtain an embedding, which is subsequently passed through a FC stack to obtain a comprehensibility score. Model is trained end-to-end.

Model	Recording-level features	LR	Val CCC	Test CCC	Test Pearson
W2Vanilla	-	0.001	0.813	0.814	0.823
1a	Accuracy	0.0003	0.821	0.813	0.827
1b		0.0007	0.818	0.81	0.824
2a	A-P contour	0.0003	0.809	0.798	0.806
2b		0.0007	0.802	0.777	0.792
3a	Prosodic miscue	0.0003	0.811	0.799	0.818
3b		0.0007	0.805	0.79	0.81
4a	Rate	0.0003	0.808	0.783	0.814
4b		0.0007	0.811	0.807	0.823
5a	Accuracy+Prosodic miscue	0.0003	**0.824**	**0.825**	**0.832**
5b	+Rate	0.0007	0.798	0.803	0.813
6a	Accuracy+A-P contour+	0.0003	0.819	0.807	0.821
6b	Prosodic miscue+Rate	0.0007	0.812	0.806	0.814

Table 5.1: Performance on concatenating various recording-level features in the W2VCat architecture. These results were obtained on extensive tuning of the learning rate and the FC stacks. The optimal configuration for FC stack 4 (through which recording-level HC features are passed) was 6 layers of 32 hidden units each.

5.2 W2VCat results

We report the performance of the W2VCat architecture, in which various HC feature sets are concatenated at the word-level and recording-level. Unless mentioned otherwise, we report performance on using the manually transcribed dataset to obtain the maximum possible improvement in performance.

5.2.1 Recording-level HC features

From table 5.1, we can observe that including accuracy, prosodic and rate features in the parallel branch (Model 5a) gives an improvement of 0.01 in test CCC over the W2Vanilla model. This aligns with our intuition since unlike acoustic features, the rest cannot be extracted from wav2vec and hence bring in complementary knowledge of canonical text and manually transcribed text. Also, the learning rate is critical; a slight increase significantly degrades performance (Model 5b). Other feature sets do not bring in any noticeable performance increment over W2Vanilla.

5.2.2 Word-level HC features

On bringing in any feature set, we see an improvement in performance over W2VAligned (row 2). However, there is no noticeable improvement over the best W2Vanilla model (row 1). This discouraging result motivated us to devise a different architecture (RNN Fusion) to exploit word-level features.

5.3 RNN Fusion results

Results for the RNN fusion are given in table 5.3. On separately training the wav2vec branch with waveforms (since W2Vanilla only requires waveforms) and the RNN branch using only non-acoustic features, followed by joint tuning of both branches, embedding-level fusion gives a slight improvement in performance over the plain W2Vanilla model. On the other hand, training the entire model from scratch gives no improvement. This indicates that a good initialisation can not only speed up training but also lead to better performance.

Model	Word-level FC features	LR	Val CCC	Test CCC	Test Pearson
W2Vanilla	-	0.001	0.813	0.814	0.823
W2VAligned	-	0.001	0.808	0.778	0.814
1a	Acoustic	0.00015	0.805	0.8	0.815
1b		5E-05	0.805	0.804	0.817
2a	Duration	0.00015	0.807	0.802	0.815
2b		5E-05	0.805	0.803	0.817
3a	LexId	0.00015	0.807	0.808	0.816
3b		5E-05	0.805	0.803	0.817
4a	LexMisc	0.00015	0.807	0.808	0.815
4b		5E-05	0.805	0.809	0.817
5a	PEDMisc	0.00015	0.806	0.803	0.813
5b		5E-05	0.805	0.803	0.816
6a	Duration+LexId	0.00015	0.806	0.805	0.817
6b	+LexMisc+PEDMisc	5E-05	0.805	0.806	0.817
7a	Acoustic+Duration+LexId	0.00015	0.808	0.802	0.815
7b	+LexMisc+PEDMisc	5E-05	0.804	0.806	0.816

Table 5.2: Performance on concatenating various word-level features in the W2VCat architecture. After tuning the hyperparameters, we find that passing the word-level features through a 4-layer stack with [96, 64, 48, 32] hidden units gives the best performance.

Features	Fusion	Pre-trained	LR	Val CCC	Test CCC	Test Pearson
HC Acoustic	-	-	0.0001	0.648	0.653	0.676
HC Non-acoustic	-	-	0.0001	0.763	0.766	0.775
HC Acoustic + Non-acoustic	-	-	0.0001	0.762	0.767	0.777
W2Vanilla	-	-	0.001	0.813	0.814	0.823
W2Vanilla + HC Acoustic	Embed.	TRUE	0.0003	0.811	0.805	0.815
		FALSE	0.0003	0.799	0.767	0.805
	Decision	TRUE	5E-05	0.792	0.795	0.805
		FALSE	5E-05	0.785	0.789	0.803
W2Vanilla + HC Non-acoustic	Embed.	TRUE	0.0003	**0.819**	**0.821**	**0.828**
		FALSE	0.0003	0.818	0.817	0.825
	Decision	TRUE	5E-05	0.817	0.800	0.824
		FALSE	5E-05	0.809	0.798	0.819
W2Vanilla + HC Acoustic + HC Non-acoustic	Embed.	TRUE	0.0001	0.818	0.82	0.827
		FALSE	0.0001	0.815	0.813	0.824
	Decision	TRUE	0.0001	0.811	0.801	0.81
		FALSE	0.0001	0.811	0.79	0.801

Table 5.3: Results of the RNN Fusion model discussed in section 5.3. Embed. and Decision refers to the two fusion techniques discussed in section 5.3. Pre-trained refers to whether the two branches are initialised with weights trained for comprehensibility; if it is False, both branches are trained from scratch.

Chapter 6

Multi-task Learning

Although HC features are computationally expensive, concatenating them with wav2vec-extracted embeddings demonstrated an improvement in performance (section 5.2). This indicates that complementary information exists between wav2vec and HC features. In order to utilise these features, we propose a new architecture inspired by the paradigm of multi-task learning (MTL). We ask the model to predict the HC features at each hierarchy. By doing so, we nudge the model to learn intermediate representations which capture such features which, in literature, are known to be helpful for the task of comprehensibility prediction. For example, knowledge of gender, age and F0 has been shown to be helpful for speech emotion recognition and hence was incorporated into an MTL framework in [38]. In [41], the authors train a self-supervised model on a variety of speech tasks such as phone recognition, KWS, ASR, speaker identification, speaker verification, emotion recognition, etc. Their findings indicate that all tasks (except speaker verification) benefit from the presence of all other auxiliary heads.

The benefit of a multi-task learning framework is two-fold. Firstly, during inference, such features are automatically predicted by the model and hence manual feature extraction is eliminated. Secondly, we can use (some) of these predictions to deliver feedback by comparing them with HC features of proficient speakers. For example, let $PV_{predicted}$ denote the the pitch variation across the utterance (one of the recording-level HC feature). If $PV_{predicted} < \mu_{PV} - 2\sigma_{PV}$ where μ_{PV} and σ_{PV} are the mean and standard deviation of pitch variation of proficient speakers, we can conclude that the child needs to be more expressive by varying the pitch.

6.1 Architecture

The architecture for MTL is given in figure 6.1. At each hierarchy, we use a stack of FC layers to predict the HC features from the wav2vec representations. These tasks are also known as auxiliary tasks. By training the model end-to-end, it is expected that the model will learn robust intermediate representations. Feature sets discussed in table 2.1 are used as targets for the auxiliary branch.

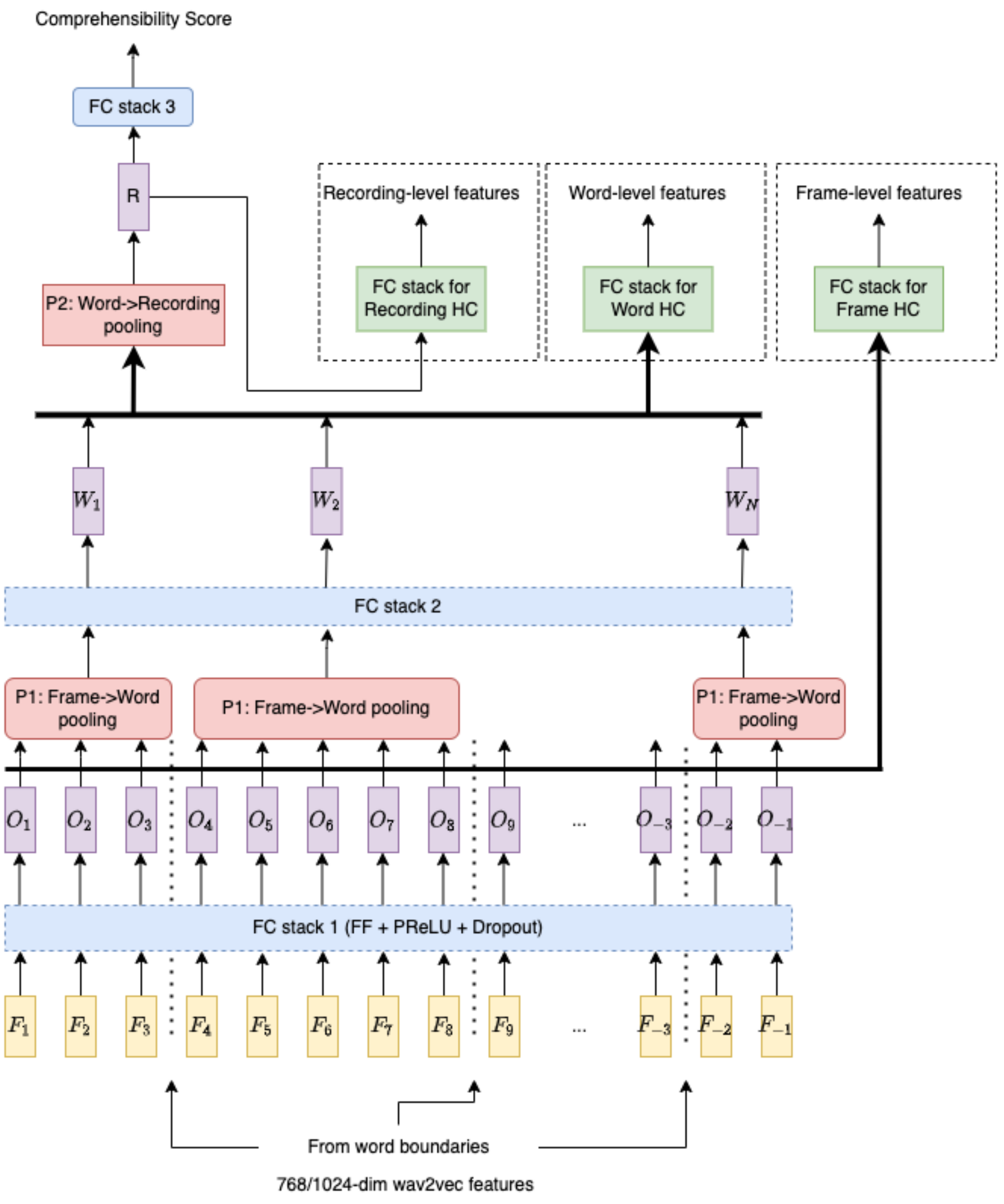
In section 5.3, we observed that on separately training the two branches of the RNN fusion architecture for comprehensibility prediction and initialising the fusion model with these weights, a slight improvement in performance is observed. On the other hand, training the model from scratch does not yield any performance benefit. A good initialisation clearly helps. Hence, we first train the MTL network for comprehensibility prediction by removing all the auxiliary tasks. Then, we initialise the model with these pre-trained weights and subsequently train the MTL model end-to-end.

6.2 Loss

Along with the comprehensibility loss, we introduce additional weighted MSE losses for each hierarchy. The final loss function is:

$$Loss = L_{compre} + \alpha_{rec}L_{rec} + \alpha_{word}L_{word} + \alpha_{frame}L_{frame} \tag{6.1}$$

where L_{compre} refers to the comprehensibility loss (discussed in section 2.3.1), α_{rec}, α_{word}



Comprehensibility Score
FC stack 3
R
Recording-level features
FC stack for Recording HC
Word-level features
FC stack for Word HC
Frame-level features
FC stack for Frame HC
P2: Word->Recording pooling
W_1
W_2
W_N
FC stack 2
P1: Frame->Word pooling
P1: Frame->Word pooling
P1: Frame->Word pooling
O_1 O_2 O_3 O_4 O_5 O_6 O_7 O_8 O_9 ... O_{-3} O_{-2} O_{-1}
FC stack 1 (FF + PReLU + Dropout)
F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9 ... F_{-3} F_{-2} F_{-1}
From word boundaries
768/1024-dim wav2vec features

Figure 6.1: The proposed MTL framework consists of fully-connected stacks at each hierarchy to extract hand-crafted features. On training the entire model end-to-end for both comprehensibility prediction and HC feature prediction, we expect the model to learn robust representations in FC stack 1 and FC stack 2. Ideally, we can experiment with more powerful deep learning models such as RNNs and CNNs (instead of FC stacks) for auxiliary tasks but we stick to FC stacks for simplicity.

and α_{frame} control the relative importance of each hierarchy and L_{rec},L_{word} and L_{frame} are the MSE losses for each hierarchy. They have been covered in equations 4.1, 4.2 and 4.3 respectively. Note that the weights can be set to 0 to drop a particular auxiliary task.

6.3 Results

The results of the MTL experiments on bringing in recording-level and word-level features (separately) are discussed in this section.

6.3.1 Recording-level auxiliary tasks

In sub-section 5.1.1, we found an improvement in performance on concatenating recording-level non-acoustic features (lexical miscue, prosodic miscue and rate). As a result, our first attempt at MTL incorporates an auxiliary branch for predicting these features. The weights α_{word} and α_{frame} are set to 0 for this experiment. Results are given in table 6.1. Despite varying the learning rate, number of layers and hidden units in the FC stack and α_{rec}, there is no improvement in performance. In fact, we see a slight decline as compared to the W2Vanilla model without any MTL auxiliary branches (row 1).

Although the FC stack hyperparameters (number of layers and hidden units) have somewhat converged to a one layer stack with around 220 units, the learning rate and α_{rec} cover a large range. Further tuning might help. Also, mean pooling, which is used to aggregate frame-level features to get a recording-level representation, could be replaced with parametric models such as GRUs or Transformers for better modelling. Lastly, we observe that the validation CCC of the MTL models is similar to W2Vanilla while the test CCC is lower than W2Vanilla. This hints towards overfitting in case of MTL.

Next, we predict recording-level acoustic features in the MTL auxiliary branch. Results are presented in table 6.2. Unfortunately, no improvement in performance is observed. Although validation performance is similar to the W2Vanilla model, a lower performance on test set again hints towards overfitting.

FC stack	α_{rec}	LR	Val CCC	Test CCC	Test Pearson
W2Vanilla	0	0.001	0.813	0.814	0.823
(209)	3.31	0.00039	0.814	0.806	0.82
(224)	2.43	0.00057	0.813	0.798	0.82
(222)	3.81	0.00088	0.813	0.788	0.821
(128)	0.65	0.00026	0.813	0.81	0.82
(248, 248)	1.43	0.00042	0.812	0.804	0.82

Table 6.1: Performance of MTL model which predicts recording-level non-acoustic HC features in an auxiliary branch. FC stack refers to the stack of fully-connected layers which is used to predict the HC features from the utterance-level wav2vec embedding.

FC stack	α_{rec}	LR	Val CCC	Test CCC	Test Pearson
W2Vanilla	0	0.001	0.813	0.814	0.823
(79, 79)	1.31	0.00019	0.816	0.804	0.819
(201, 201)	2.30	0.00014	0.816	0.807	0.819
(103, 103)	0.92	0.00018	0.815	0.808	0.821
241	3.33	0.00046	0.815	0.809	0.82
(144, 144)	0.59	0.00014	0.814	0.806	0.819

Table 6.2: Performance of MTL model which predicts recording-level acoustic HC features in an auxiliary branch.

Pre-word stack	α_{word}	LR	Val CCC	Test CCC	Test Pearson
W2Vanilla	0	0.001	0.813	0.814	0.823
W2VAligned	0	0.001	0.808	0.778	0.814
32	0.87	0.00036	0.81	0.796	0.811
32	0.84	0.00036	0.81	0.794	0.813
32	0.76	0.00047	0.81	0.791	0.815
(16, 16, 16)	1.40	0.00017	0.809	0.808	0.815
(32, 16)	0.21	0.00016	0.809	0.805	0.814

Table 6.3: Results for MTL with word-level acoustic HC features as targets in an auxiliary branch. Word-level features are passed through a stack of fully-connected layers before concatenation with acoustic representation; the hidden units of each layer in this stack is mentioned in column *Pre-word stack.*

6.3.2 Word-level auxiliary tasks

In this subsection, we report results on setting $\alpha_{rec} = \alpha_{frame} = 0$ and varying α_{word}. We experiment with 2 feature sets as MTL targets: acoustic and non-acoustic (which includes lexical, miscue, duration and PEDMiscue).

From table 6.3, we can infer that no improvement in performance is observed. Again, mean pooling to obtain a word-level representation from frame-level representations could be the culprit. For example, pitch variation across a word is an important feature which is difficult to capture using mean pooling.

In table 6.4, performance of models trained to additionally predict word-level non-acoustic features is discussed. Again, no improvement in performance on using non-acoustic features as MTL targets. However, hyperparameters such as learning rate and α_{word} are yet to converge, indicating that further tuning might help.

Pre-word stack	α_{word}	LR	Val CCC	Test CCC	Test Pearson
W2Vanilla	0	0.001	0.813	0.814	0.823
W2VAligned	0	0.001	0.808	0.778	0.814
32	0.87	0.00028	0.808	0.798	0.812
32	0.11	0.00068	0.808	0.78	0.815
32	0.55	0.00071	0.807	0.794	0.814
(32, 32)	0.91	0.00060	0.807	0.792	0.813
(32, 32)	0.60	0.00099	0.805	0.794	0.812

Table 6.4: Results for MTL with word-level non-acoustic HC features as targets in an auxiliary branch.

6.3.3 Concluding remarks

Since we did not observe any improvement in performance for any of the feature sets, we also tried training the entire model from scratch (instead of initialising it with weights pre-trained for comprehensibility). However, it did not yield any improvement.

A possible direction to explore would be to include both recording and frame-level branches ($\alpha_{word} \neq 0, \alpha_{rec} \neq 0$). In particular, the recording-level branch could benefit greatly from this since a number of recording-level features are aggregates of word-level features. However, training the network becomes tricky since we not only have two heads to tune but the relative importance of the two (and with respect to comprehensibility) also needs to be tuned (α_{rec} and α_{word}). Also, for simplicity purposes, we are currently using MSE as the loss function for all features. However, binary features such as prosodic events, POS tags, etc. could benefit from losses which are designed for classification such as binary cross entropy loss.

Although the proposed architecture can incorporate frame-level MTL, we first experimented with word-level and recording-level MTL heads since we believe frame-level predictions will be too noisy and fine-grained for the model to extract anything meaningful. Moreover, the probing experiments suggest that the model does a decent job at capturing most frame-level contours as compared to the other two hierarchies (check subsection 4.4.3).

Chapter 7

Summary

In summary, we have successfully replaced a RFC model operating on HC features with a deep learning alternative. Our baseline deep learning networks, operating on HC features, was already found to outperform the RFC. We then introduced our first wav2vec-based architecture which only requires waveform as input. It outperformed the deep learning baselines, which operate on features derived after extensive hand-engineering. Moreover, the end-to-end nature of the deep learning model makes it easy to train, as opposed to multiple independent components of the feature extraction framework.

We then focus on inspecting the wav2vec representations. We do so by adding linear probes at each hierarchy and asking the model to predict HC features, which are interpretable. At the frame-level, we discover that it is possible to extract most contours with high correlation. Pitch contour shapes at the word-level are difficult to extract, which is expected since our pooling technique for computing word-level representations from frame-level representations is trivial mean pooling. Word-level lexical features such as POS tags and miscue tags are difficult to extract since the model has no knowledge of the canonical text. At the recording-level, rate-based features and prosodic aggregates can be extracted from the tuned wav2vec model.

Next, we tested for complementary information between HC features and wav2vec by incorporating the HC features in the wav2vec architecture. We observed an improvement when non-acoustic recording-level features were concatenated with the wav2vec representation. This is in line with our expectations since wav2vec has no knowledge of canonical text and the ASR decoded text. On the other hand, word-level features did not improve performance. We also tried out a different fusion technique for fusing the word-level lexical information and wav2vec acoustic information. On initialising both branches of the architecture with weights trained for the task of comprehensibility, we observed a slight improvement in performance.

Finally, we discussed a multi-task learning framework which incorporates auxiliary branches for predicting HC features at various hierarchies. Although we did not observe any improvement in performance, we believe it is a scalable architecture which might benefit from better hyperparameter tuning, pooling strategies and more data.

7.1 Final results

A summary of the performances of all models is given in table 7.1. On simply replacing RFC with a MLP, we see an improvement in the range 0.015-0.02 for Test Pearson. W2Vanilla, without any knowledge of canonical and ASR decoded text, demonstrated an improvement of 0.025 in Test CCC over the recording-level MLP trained on both lexical and acoustic feature sets. Next, we experimented with various ways of bringing in HC features in the W2Vanilla architecture. Our best system (W2VCat) improves over the RFC baseline by 0.065 (Test Pearson). However, the improvement over W2Vanilla is only 0.01. Furthermore, MTL did not lead to any performance improvement. Thus, W2VCat is our **best** performing system.

7.2 Future Work

We plan to pursue the following directions in the future:

- For all wav2vec architectures, the transformer is frozen due to lack of computational resources. It would be interesting to check if fine-tuning them further leads to any improvement, especially in a low resource setting (since we have only 10 hours of speech).

Model	HC Features	Dataset	Val CCC	Test CCC	Test Pearson
RFC	Recording (All)	ASR	-	0.717	0.760
		FA	-	-	0.777
MLP		ASR	0.769	0.767	0.777
		FA	0.793	0.789	0.796
RNN	Word (All)	ASR	0.738	0.741	0.751
		FA	0.762	0.767	0.777
W2Vanilla	-	-	0.813	0.814	0.823
W2VAligned	-	ASR	0.804	0.781	0.815
	-	FA	0.808	0.778	0.814
W2VCat	Recording (All except A-P contour)	ASR	0.814	0.797	0.822
		FA	**0.824**	**0.825**	**0.832**
RNN Fusion (Embedding)	Word (All except A-P contour)	ASR	0.813	0.815	0.823
		FA	0.818	0.82	0.827
MTL	Recording (A-P contour)	FA	0.816	0.804	0.819
	Recording (All except A-P contour)	FA	0.814	0.806	0.82
	Word (A-P contour)	FA	0.81	0.796	0.811
	Word (All except A-P contour)	FA	0.808	0.798	0.812

Table 7.1: Summary of performances of all the proposed architectures, along with the baseline.

- To alleviate the issue of a limited dataset, data augmentation techniques such as SpecAugment [42] can be explored. It is important to note that techniques which change the speech rate are applicable for tasks such as ASR but not for comprehensibility since speech rate affects the prediction. In [43], new samples are generated by simply extracting a random slice of an utterance; since it is an utterance-level task, the label is preserved.

- Domain shift: While Wav2vec2.0 is pre-trained on LibriSpeech which consists of audiobook recordings, our dataset consists of children's recordings. Studies such as [44] have studied the effect of domain shift between pre-training, fine-tuning and test dataset. Unsurprisingly, their key finding is that adding test-domain data to the pre-training phase helps. Hence, it would be interesting to explore if pre-training wav2vec on a large corpus of children's dataset (potentially a collection of various small corpora) improves performance.

- Transfer learning: Emotion recognition is a closely related task since prosody is a crucial acoustic factor for predicting the utterance-level score. To alleviate the issue of a limited dataset, we can improve the low-level feature extractors (such as the CNN of wav2vec) by training it for emotion recognition and initialising our comprehensibility model with these learned weights.

- Dataset size: Since obtaining comprehensibility ratings is an expensive endeavour (both time and skill-wise), it would be interesting to analyse the performance of our wav2vec-based model on varying the size of the dataset. Since most other approaches train an entire network from scratch, their performance might drop significantly in a low-resource setting as compared to a wav2vec-based model (or any pre-trained model).

- Beyond wav2vec: After Wav2vec2.0, many closely related self-supervised models have been proposed. For example, XLS-R [45] is trained on about 500,000 hours of publicly available speech audio in 128 languages. Since our dataset consists of L2 English speakers, we might benefit from a cross-lingual model. Data2vec [46] proposes a general framework for speech, vision and language and has demonstrated superior performance over Wav2vec2.0 on low-resource ASR. It would be interesting to explore the performance of these models on our task.

- Wav2vec has demonstrated excellent performance for ASR in literature [6, 47]. Since our task also involves an ASR component, it would be beneficial to incorporate a separate branch for ASR. This not only has the potential to improve ASR performance but also reduce the overall computation time during deployment since the wav2vec extracted features could be used for both ASR and Comprehensibility.

- A self-supervised framework is discussed next.

Wav2vec2.0 is trained using contrastive loss for predicting masked representations. It is problem agnostic by design. However, from literature, we know of several acoustic properties of speech which are crucial for comprehensibility prediction **and** can be derived from the signal without any human labelling. Hence, we propose a self-supervised model which is trained to predict such properties, thereby biasing the model towards such known features. Although obtaining comprehensibility ratings is time consuming since it requires skilled teachers, collecting speech recordings is much easier. Currently, apart from 10 hours of data labelled for comprehensibility, we have an additional 20 hours of unlabelled children's speech data. We propose an architecture which can be exploit this unlabelled data in a self-supervised fashion. It is inspired by PASE [48, 49] which is briefly discussed below.

PASE consists of a convolutional encoder which accepts waveform as input and outputs frame-level representations. These representations are fed to various workers. The worker targets can be extracted from the waveform itself (hence the term **self**-supervised). Examples of worker tasks include MFCC prediction, waveform sample recovery and Log Power Spectrum prediction.

The shared encoder is forced to extract meaningful representations in order to solve all the worker tasks which, from domain knowledge, are known to be important characteristics of speech for a number of downstream tasks. PASE was pre-trained on about 10 hours of speech, consisting of 15 second recordings from 2848 speakers. For speech emotion recognition, when the encoder was further fine-tuned along with the classifier layer, they observed an improvement of 6% in classification accuracy over standard features such as MFCCs and Filterbanks. As expected, ablation studies demonstrated a drop in performance on discarding the *Prosody* worker (which consists of predicting four frame-level features - the interpolated logarithm of the fundamental frequency, voiced/unvoiced probability, zero-crossing rate, and energy). Also, simply training the PASE architecture in a supervised fashion for the downstream task led to a

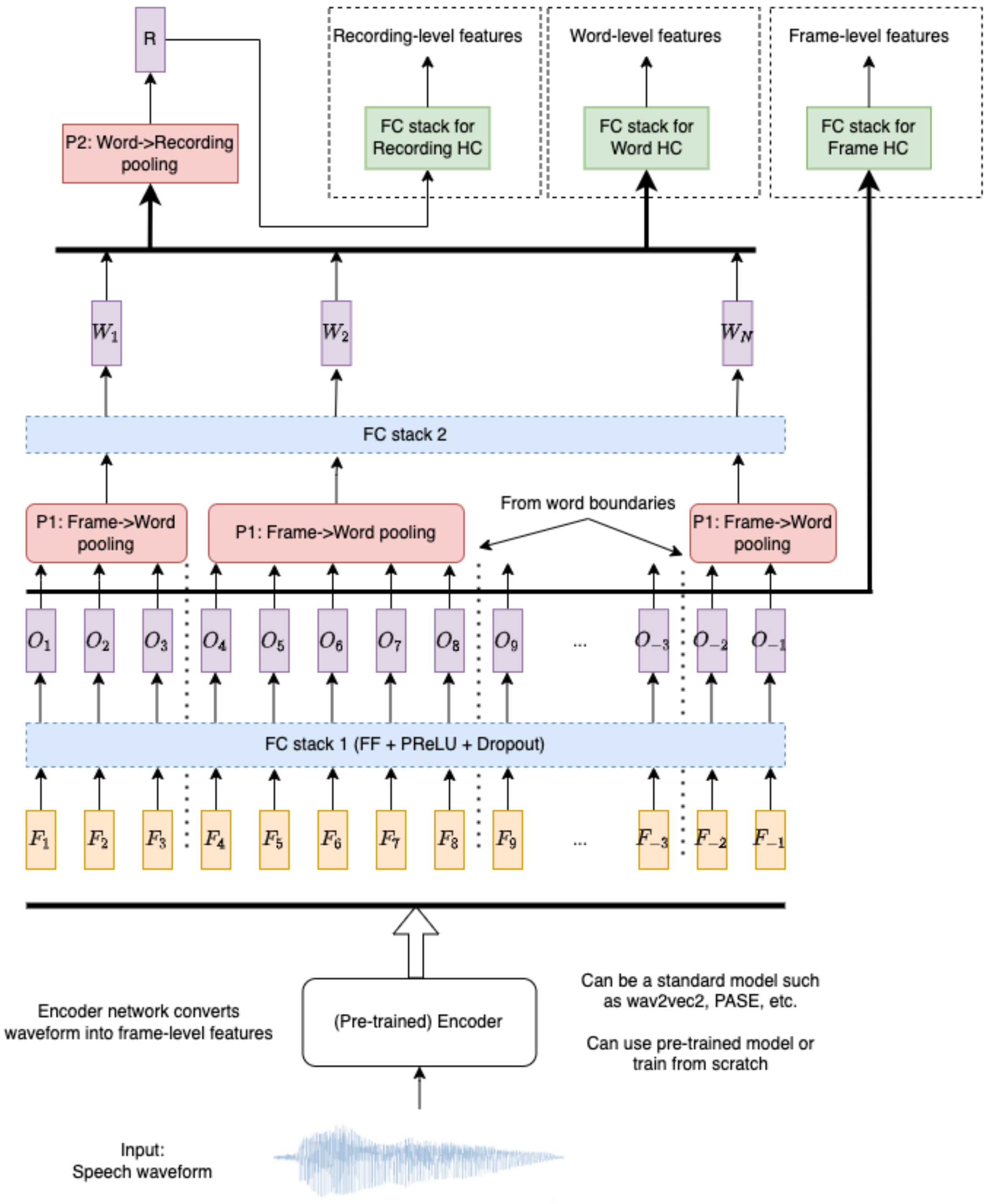

R
P2: Word->Recording pooling
Recording-level features
FC stack for Recording HC
Word-level features
FC stack for Word HC
Frame-level features
FC stack for Frame HC
W_1
W_2
W_N
FC stack 2
P1: Frame->Word pooling
P1: Frame->Word pooling
From word boundaries
P1: Frame->Word pooling
O_1 O_2 O_3 O_4 O_5 O_6 O_7 O_8 O_9 ... O_{-3} O_{-2} O_{-1}
FC stack 1 (FF + PReLU + Dropout)
F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9 ... F_{-3} F_{-2} F_{-1}
Encoder network converts waveform into frame-level features
(Pre-trained) Encoder
Can be a standard model such as wav2vec2, PASE, etc.
Can use pre-trained model or train from scratch
Input: Speech waveform

Figure 7.1: Proposed architecture which can be trained in a self-supervised fashion without comprehensibility labels.

4% decline in performance. It indicates the effectiveness of pre-training on a number of worker tasks before fine-tuning for the task at hand.

Motivated by PASE, we propose an architecture in figure 7.1, which is a slightly modified version of the MTL architecture (figure 6.1). At each hierarchy, we ask the model to predict HC features which are known to be useful for comprehensibility prediction. We can use any standard deep learning model as the encoder (Wav2vec2.0, PASE, etc.). Furthermore, we can initialise it with pre-trained weights (e.g. Wav2vec2.0 pre-trained on Librispeech) so as to speed up training. Since PASE was trained on just 10 hours of data while we currently have around 30 hours of unlabelled children's speech, dataset size will not be a major bottleneck.

Another motivation for the self-supervised framework is the issue of domain mismatch. The Wav2vec2.0 model, which is being currently used for extracting frame-level embeddings, was pre-trained on LibriSpeech [50], a dataset consisting of audiobook recordings (by presumably adult speakers since it's recordings are sourced from LibriVox, an open-source project). Thus, there is a major domain mismatch between the pre-training dataset and our test dataset which consists of L2 children speakers. Naturally, such a mismatch degrades performance [44]. For example, due to differences in the pitch range of adults and children, the CNN encoder pre-trained on adult speech may perform poorly when it comes to pitch extraction on children's speech. Hence, it would be beneficial to train our own self-supervised model on children's speech from scratch.

This page was intentionally left blank.

Appendix A

Wav2vec2.0 for PED

In stage 1 of our work, we proposed a CRNN framework for predicting prosodic events (prominence and phrase boundary). It consisted of a convolutional neural network for extracting features from waveform segments and a Bidirectional GRU for modelling context across words. Using multi-task learning and Sinc-based convolutions, we managed to outperform a BGRU trained on hand-crafted features [3]. A summary of the results is given in table A.1.

The best result without any hand-crafted features (row 3 in table A.1) used Sinc filters and benefited from conditioning of the prominence prediction on the phrase boundary prediction.

A.1 Wav2vec2.0-based architecture

Instead of extracting features from raw waveform, we replace them with frame-level wav2vec embeddings. We can then use standard deep learning blocks such as CNNs or RNNs to pool the word-level embedding chunks and pass it through a sequence encoder to obtain prosodic event predictions. A similar approach was proposed in [51]: CNNs are used to extract information from 32 low-level descriptors (such as F0, voicing probability, MFCCs, etc.) and predict a

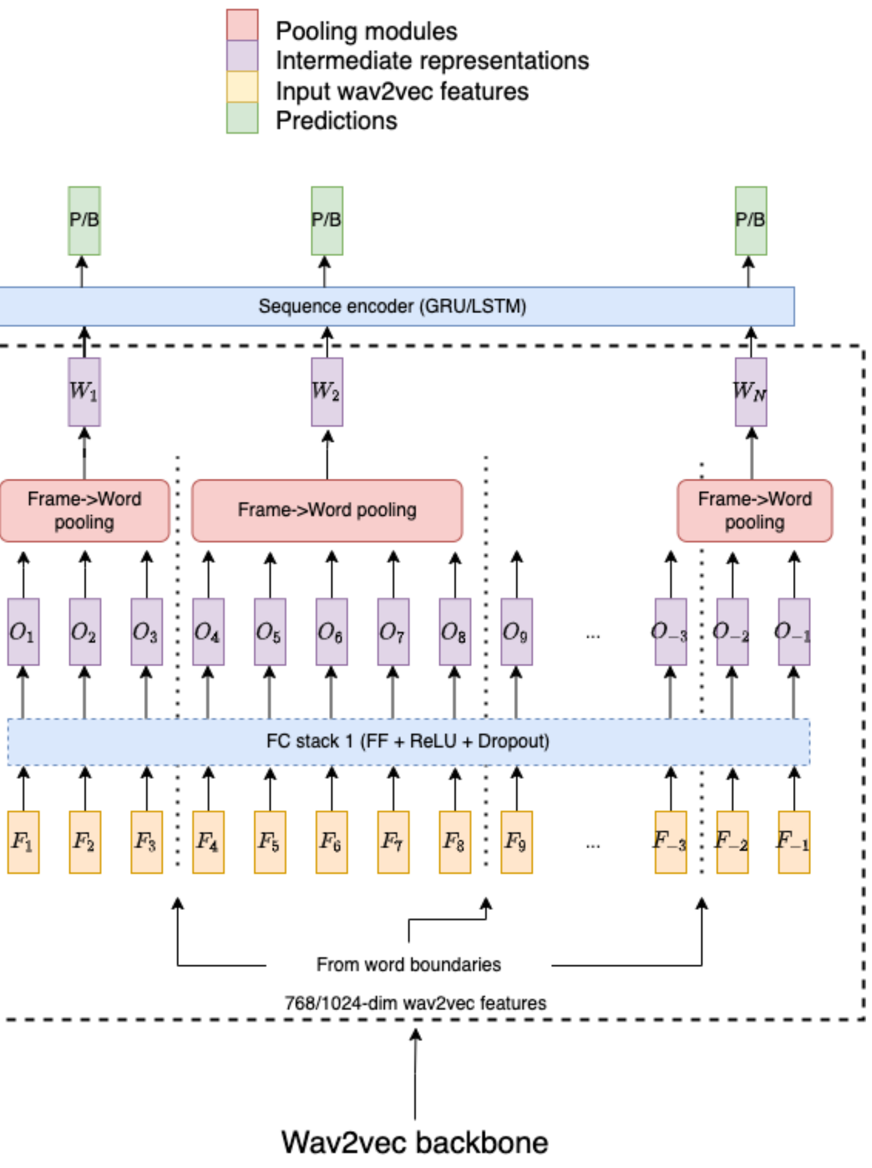

Pooling modules
Intermediate representations
Input wav2vec features
Predictions
P/B
P/B
P/B
Sequence encoder (GRU/LSTM)
W_1
W_2
W_N
Frame->Word pooling
Frame->Word pooling
Frame->Word pooling
O_1 O_2 O_3 O_4 O_5 O_6 O_7 O_8 O_9 ... O_{-3} O_{-2} O_{-1}
FC stack 1 (FF + ReLU + Dropout)
F_1 F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9 ... F_{-3} F_{-2} F_{-1}
From word boundaries
768/1024-dim wav2vec features
Wav2vec backbone

Figure A.1: Proposed wav2vec-based architecture for prosodic event detection.

Task	Model	HC features	MTL	Test Pearson
Prominence	GRU	A34	N	0.725
	SincConv+GRU	-	N	0.721
	SincConv+GRU	-	Y	**0.74**
	SincConv+GRU	A34+A27	Y	0.757
	SincConv+GRU	A34+A27+GloVe	Y	0.8
Boundary	SincConv+GRU	-	N	0.887
	SincConv+GRU	-	Y	**0.894**
	SincConv+GRU	A27+GloVe	Y	0.927

Table A.1: Summary of PED results reported in our previous work [3]. The performance of the best system without hand-crafted features is highlighted in bold.

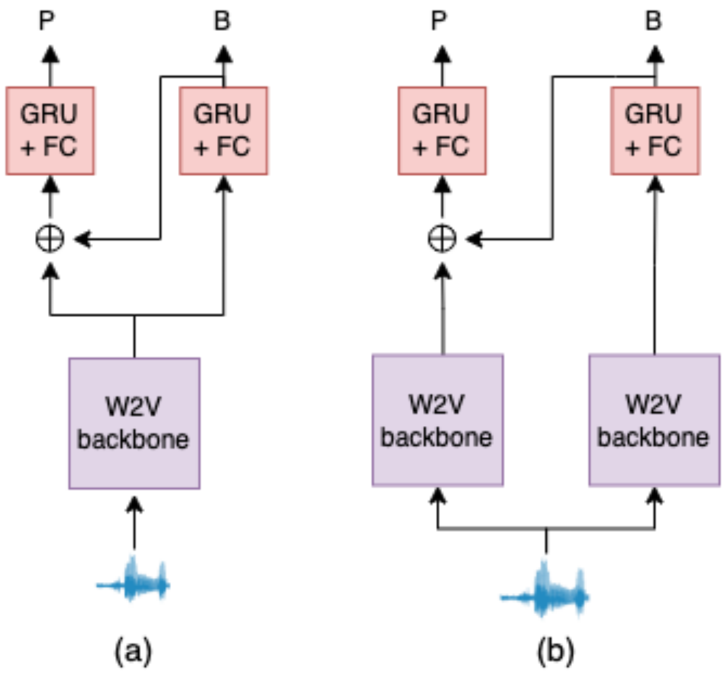


Figure A.2: The two conditioning architectures for MTL. In (a), the wav2vec backbone is shared across the two tasks. In (b), we have separate wav2vec backbone so that the model can extract different task-specific features.

Task	Pooling	Stack 1	MTL	Test Pearson
Prominence	Mean	[512, 256]	N	0.745
	Mean	[512, 256]	Y (shared backbone)	0.737
	Mean	[512, 256]	Y (separate backbones)	0.744
	CNN	[512, 256]	N	0.387
	CNN	-	N	**0.759**
Boundary	Mean	[512, 256]	N	0.901
	CNN	[512, 256]	N	**0.906**
	CNN	-	N	0.906

Table A.2: Performance of the proposed wav2vec-based architecture for PED. MTL refers to the two conditioning architectures shown in figure A.2.

word-level label. Our proposed architecture can be found in figure A.1. We also experiment with the MTL architecture shown in figure A.2. We try out both configurations: using a shared wav2vec backbone or using separate wav2vec backbone for prominence and phrase boundary. In both cases, we condition the prominence prediction on the phrase boundary prediction.

A.2 Results

We use the same sequence encoder as we did in our previous work: a 2-layer, 256-dimensional bidirectional GRU. We experiment with 2 pooling modules: mean pooling and CNN filtering followed by max pooling. In mean pooling, we simply average the frame-level representations to obtain a word-level representation W_k. For CNN, we use two kernel widths: 25 and 51, which give a receptive field of around 500 ms and 1 second respectively. For each kernel, we have a 2-layer CNN with 128, followed by 64 filters. For each kernel, the output of the CNN is max-pooled across time to obtain a single word-level embedding. The output embeddings of both the kernel networks are concatenated to obtain the final word-level representation W_k.

We use the same folds and training methodology as we did in stage 1. Performance is reported in table A.2.

Results indicate that the wav2vec-based model which uses mean pooling and without any FC layers (row 5 in table A.2) gives a noticeable improvement (0.02 absolute Pearson corr.) for prominence over the Sinc-based MTL architecture without any hand-crafted features (row 3 in table A.1). For phrasing, we see an improvement of 0.01 in Test Pearson. Note that we did not tune the hyperparameters (learning rate, batch size, CNN kernel widths, number of filters, etc.) of the wav2vec architecture. Hence, it is reasonable to expect some further improvement on tuning these critical hyperparameters.

This result is important from the perspective of deployment: wav2vec features will be extracted for comprehensibility prediction. In that case, we can do away with the Sinc-based CNN model and simplify the pipeline by using the same wav2vec features for both comprehensibility and PED. This will reduce compute time. Moreover, we could explore joint optimisation of the PED and Comprehensibility networks since both networks could benefit from each other.

Appendix B

Word-level pooling

In W2Aligned, we used mean pooling to obtain a word-level representation from frame-level representations and word alignments. Ideally, deep learning models such as CNN, RNNs, etc. can learn intricate patterns and perform much better than trivial mean pooling. However, we run into an implementation issue for more sophisticated pooling techniques as discussed below.

GPUs are extremely fast at computing matrix multiplications. All deep learning components can be broken down into a series of matrix multiplications. As a result, the deep learning revolution has been significantly driven by increases in GPU performance. Also, unlike images, each sample in speech datasets tend to be of different lengths. As a result, they need to be padded in order to construct a tensor which can be subsequently fed to a GPU.

With this background, let us now come to the crucial difference between mean pooling and other deep learning architectures.

In the example depicted in figure B.1, we have N=3 words in the utterance consisting of F1, F2 and F3 frames (Total number of frames = F1 + F2 + F3 = F). The wav2vec feature matrix is of dimensions (D, F) where D is the embedding dimension.

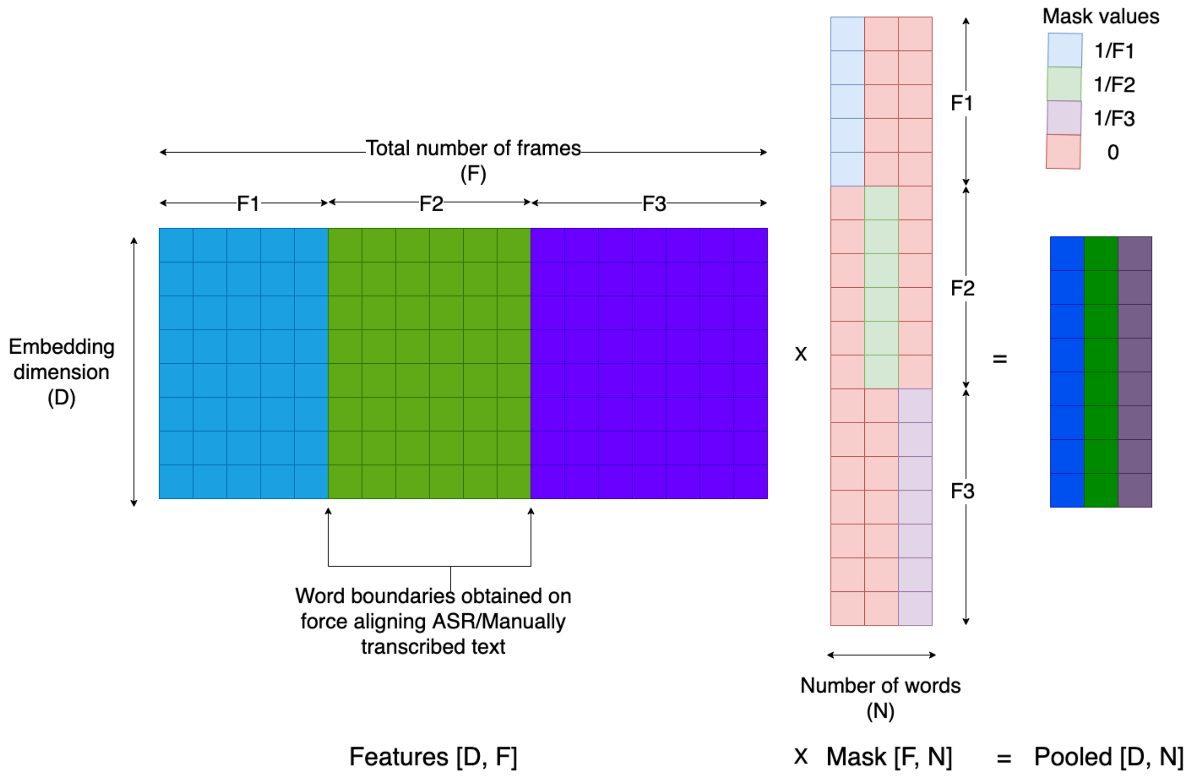


Figure B.1: Implementation of mean pooling to aggregate frame-level features and obtain word-level representation. The example consists of 3 words with F1, F2 and F3 number of frames.

Mean pooling can be implemented with a giant matrix multiplication. To do so, we construct a mask of dimensions (F, N) as depicted in the figure. The first column has value 1/F1 in the first F1 rows and 0 otherwise. The second column has 1/F2 in rows [F1, F1+1, ..., F1+F2] and 0 elsewhere. Similarly for the third column. We can obtain the mean pooled representation by multiplying the feature matrix with the mask, as depicted in figure B.1.

On the other hand, for any other pooling mechanism, we need to first split the utterance using the word alignments into chunks of frame-level features. Then, they need to be padded in order to construct a batch. We can now pass this batch to a standard deep learning block such as RNN or CNN to obtain a pooled representation. Please see figure B.2 for a visual explanation of the same.

The training procedure is significantly slowed down by this procedure, despite using vectorised PyTorch operations such as *torch.split()* instead of loops[1] for splitting the utterance into word-level chunks. Moreover, say an utterance consists of N words with frame lengths F1,

[1]Loops can cause a major bottleneck if not implemented efficiently on a GPU.

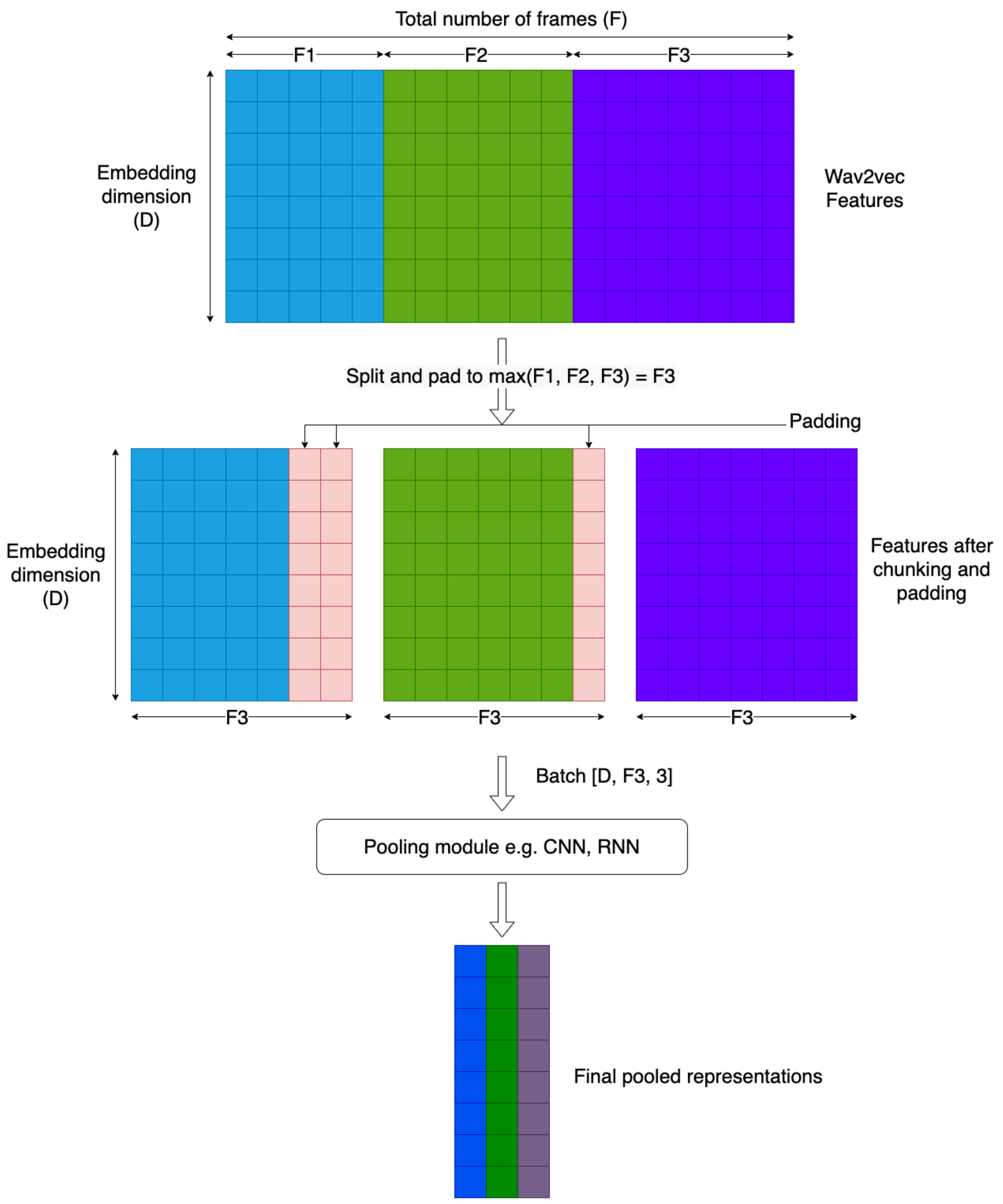

Total number of frames (F)
F1
F2
F3
Embedding dimension (D)
Wav2vec Features
Split and pad to max(F1, F2, F3) = F3
Padding
Embedding dimension (D)
Features after chunking and padding
F3
F3
F3
Batch [D, F3, 3]
Pooling module e.g. CNN, RNN
Final pooled representations

Figure B.2: Implementation of a generic pooling to aggregate frame-level features and obtain word-level representation.

F2 ,..., FN. As depicted in figure B.2, we need to pad the chunks to max(F1, F2, ..., FN). If a particular utterance consists of even one long word, each chunk will need to be padded to construct a batch. The unavoidable padding will lead to a significant increase in the number of computations in the pooling module.

While W2VAligned with mean pooling takes around 1 hour to train, replacing the word-level pooling module with a CNN or a RNN takes around 10 hours. On running a couple of experiments with standard CNN and RNN hyperparameters, no performance increase was observed. It is difficult to conclude the effectiveness of such pooling without extensive tuning of the hyperparameters. However, due to time constraints, we did not pursue it further.

Appendix C

Timing Analysis

In this appendix, we analyse the time taken by W2Vanilla to generate prediction(s). This is important from the perspective of deployment. Techniques such as pruning of weights and quantisation of weights can be explored in case the model fails to meet the timing criterion.

The current pipeline can be broken down into 2 main steps:

1. Wav2vec feature extraction: in this stage, we read the .wav file, load the Wav2vec2.0 model into memory and compute a forward pass through the CNN and transformer stack of wav2vec to generate the frame-level representations (red bars in all subsequent plots).

2. Score prediction and ensembling: The wav2vec features are fed to W2Vanilla model, which consists of FC stacks and pooling mechanisms. For each utterance, we compute 30 scores from the 30 trained models[1] and average it out to obtian the final prediction (blue bars in all subsequent plots).

We experiment with both *wav2vec2-base* and *wav2vec2-large* since the two differ in the sizes of the transformer and the embedding dimensionality which significantly affect the time

[1]for each of the 6 test sets, we have 5 models trained in a CV manner.

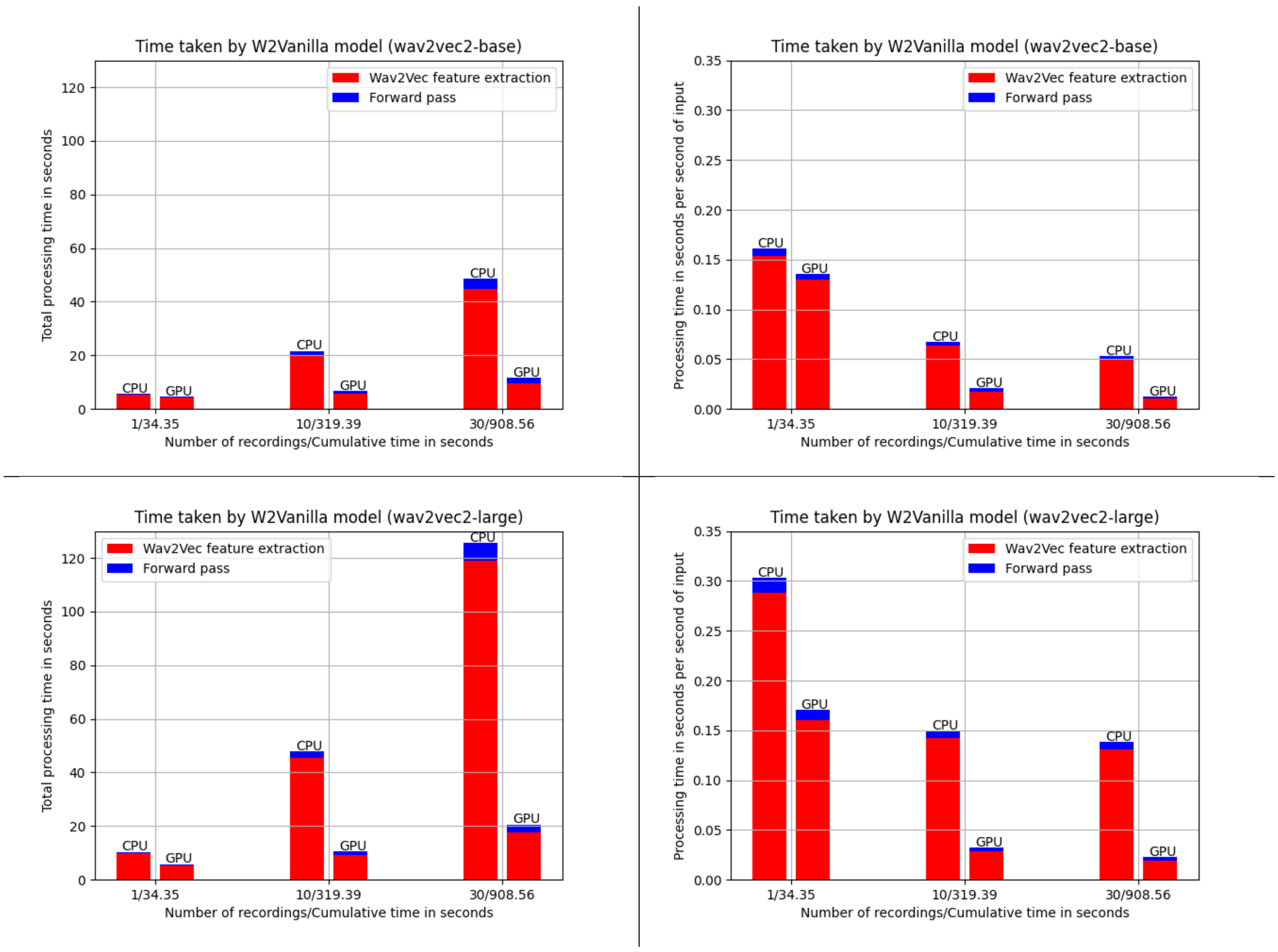


Figure C.1: Time taken by base and large models on varying duration of the recordings. On X-axis, we have an increasing number of recordings. Y-axis reports either the total time taken (left half) or time taken per second of audi input (right half). Results on using the base/large models are reported in the first/second row respectively.

taken to compute features. Also, performance is reported for both CPU and GPU, whose specifications are given below:

- GPU specifications: NVIDIA GeForce GTX 1080 Graphics Cards with 8 GB GDDR5X memory and 2560 CUDA cores.
- CPU specifications: Processor - Intel(R) Core(TM) i9-9940X CPU @ 3.30GHz. 14 CPU cores, 28 siblings (hyperthreading.; CPU-only experiments used 10 cores (specified using *taskset -c 0-10*). Total RAM: 128 GB

The results are given in figure C.1. First, let us discuss the base model and stick to CPU performance. Total time taken (left plot in row 1) seems to scale approximately linearly with the total duration of the recordings (clear with the CPU numbers). This is expected since number of frame-level embeddings which need to be extracted from wav2vec linearly increases with the duration of the utterance. The plot on the right, which depicts the time taken per second of input recording, seems to decrease upto a certain point and then saturate. This can be potentially explained by two important calls in the wav2vec feature extraction stage: loading the wav2vec model into memory and computing the forward pass. The former is independent of the duration of the recording while the latter will scale linearly with duration as discussed previously. In this case, time taken for wav2vec feature extraction can be modelled as $Ax + B$ where A depends on the configuration of the wav2vec model, x is the duration of the input and B is the time taken to load the model into memory for inference. This hypothesis can be further tested by reporting the time taken by the two stages separately.

On moving from base to large, we observe an approximate doubling of the total time taken. This can be explained by the doubling of the number of transformer encoders as we move from base (12 layers) to large (24 layers), which are responsible for the bulk of the computation involved in generating the embeddings.

In all the plots, we can observe that generating the final score by passing the wav2vec features through the FC stacks and pooling, followed by 30-fold ensembling, forms a minuscule fraction of the total time. This allows us to use more sophisticated models in the architecture which follows wav2vec features.

For all settings, the GPU consistently outperforms the CPU by a large margin as expected. The speedup is significant. For example, for *wav2vec2-large*, it provides an approximate 6x speedup over the CPU.

References

[1] Kamini Sabu. *Automatic Assessment of Fluency in Children's Oral Reading using Prosody Modeling*. PhD thesis, Indian Institute of Technology Bombay, Mumbai, India, Submitted on 28th February 2022.

[2] Leonardo Pepino, Pablo Riera, and Luciana Ferrer. Emotion recognition from speech using wav2vec 2.0 embeddings. *arXiv preprint arXiv:2104.03502*, 2021.

[3] Mithilesh Vaidya, Kamini Sabu, and Preeti Rao. Deep learning for prominence detection in children's read speech. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8157–8161, 2022.

[4] K Neuendorf. The pearson correlation coefficient vs. lin's concordance coefficient.

[5] Chaya Bakshi. Random forest regression, 2020.

[6] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[7] ASER Centre. Aser: The annual status of education report (rural) 2016. Technical report, 2017.

[8] JK Torgesen. Catch them before they fall: Identification and assessment to prevent reading failure in young children (on-line). *National Institute of Child Health and Human Development. Available: ldonline. org. ld_indepth/reading/torgesen_catchthem. html*, pages 1–15, 1998.

[9] Melanie R Kuhn, Paula J Schwanenflugel, and Elizabeth B Meisinger. Aligning theory and assessment of reading fluency: Automaticity, prosody, and definitions of fluency. *Reading research quarterly*, 45(2):230–251, 2010.

[10] Jan E Hasbrouck and Gerald Tindal. Curriculum-based oral reading fluency norms for students in grades 2 through 5. *Teaching Exceptional Children*, 24(3):41–44, 1992.

[11] Sheila W Valencia, Antony T Smith, Anne M Reece, Min Li, Karen K Wixson, and Heather Newman. Oral reading fluency assessment: Issues of construct, criterion, and consequential validity. *Reading Research Quarterly*, 45(3):270–291, 2010.

[12] Jackson J Liscombe. *Prosody and speaker state: paralinguistics, pragmatics, and proficiency*. Columbia University, 2007.

[13] Zhou Yu, Vikram Ramanarayanan, David Suendermann-Oeft, Xinhao Wang, Klaus Zechner, Lei Chen, Jidong Tao, Aliaksei Ivanou, and Yao Qian. Using bidirectional lstm recurrent neural networks to learn high-level abstractions of sequential features for automated scoring of non-native spontaneous speech. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 338–345. IEEE, 2015.

[14] Lei Chen, Jidong Tao, Shabnam Ghaffarzadegan, and Yao Qian. End-to-end neural network based automated speech scoring. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6234–6238. IEEE, 2018.

[15] Manraj Singh Grover, Yaman Kumar, Sumit Sarin, Payman Vafaee, Mika Hama, and Rajiv Ratn Shah. Multi-modal automated speech scoring using attention fusion, 2020.

[16] Yao Qian, Patrick Lange, Keelan Evanini, Robert Pugh, Rutuja Ubale, Matthew Mulholland, and Xinhao Wang. Neural approaches to automated speech scoring of monologue and dialogue responses. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8112–8116, 2019.

[17] Klaus Zechner, Derrick Higgins, Xiaoming Xi, and David M Williamson. Automatic scoring of non-native spontaneous speech in tests of spoken english. *Speech Communication*, 51(10):883–895, 2009.

[18] Hannah Muckenhirn, Vinayak Abrol, Mathew Magimai-Doss, and Sébastien Marcel. Understanding and visualizing raw waveform-based cnns. In *Interspeech*, pages 2345–2349, 2019.

[19] Sabrina Stehwien, Antje Schweitzer, and Ngoc Thang Vu. Acoustic and temporal representations in convolutional neural network models of prosodic events. *Speech Communication*, 125:128–141, 2020.

[20] Haiyang Xu, Hui Zhang, Kun Han, Yun Wang, Yiping Peng, and Xiangang Li. Learning alignment for multimodal emotion recognition from speech. *arXiv preprint arXiv:1909.05645*, 2019.

[21] Guang Shen, Riwei Lai, Rui Chen, Yu Zhang, Kejia Zhang, Qilong Han, and Hongtao Song. Wise: Word-level interaction-based multimodal fusion for speech emotion recognition. In *INTERSPEECH*, pages 369–373, 2020.

[22] Mariana Rodrigues Makiuchi, Kuniaki Uto, and Koichi Shinoda. Multimodal emotion recognition with high-level speech and text features. *arXiv preprint arXiv:2111.10202*, 2021.

[23] Kamini Sabu and Preeti Rao. Prosodic event detection in children's read speech. *Computer Speech Language*, 68:101200, 2021.

[24] Sheida White, John Sabatini, Bitnara Jasmine Park, Jing Chen, Jared Bernstein, and Mengyi Li. The 2018 naep oral reading fluency study. nces 2021-025. *National Center for Education Statistics*, 2021.

[25] Shreeharsha B. S. Acoustic models for speech recognition in children's reading miscue detection. 2021.

[26] Fabien Ringeval, Björn Schuller, Michel Valstar, Shashank Jaiswal, Erik Marchi, Denis Lalanne, Roddy Cowie, and Maja Pantic. Av+ ec 2015: The first affect recognition challenge bridging across audio, video, and physiological data. In *Proceedings of the 5th international workshop on audio/visual emotion challenge*, pages 3–8, 2015.

[27] I Lawrence and Kuei Lin. A concordance correlation coefficient to evaluate reproducibility. *Biometrics*, pages 255–268, 1989.

[28] Vedhas Pandit and Björn Schuller. The many-to-many mapping between the concordance correlation coefficient and the mean square error. *arXiv preprint arXiv:1902.05180*, 2019.

[29] Bagus Tris Atmaja and Masato Akagi. Evaluation of error-and correlation-based loss functions for multitask learning dimensional speech emotion recognition. In *Journal of Physics: Conference Series*, volume 1896, page 012004. IOP Publishing, 2021.

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*, San Diego, CA, 2015.

[31] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '19, page 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery.

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[33] Jonathan Boigne, Biman Liyanage, and Ted Östrem. Recognizing more emotions with less data using self-supervised transfer learning. *arXiv preprint arXiv:2011.05585*, 2020.

[34] Manon Macary, Marie Tahon, Yannick Estève, and Anthony Rousseau. On the use of self-supervised pre-trained acoustic and linguistic features for continuous speech emotion recognition. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 373–380. IEEE, 2021.

[35] Nik Vaessen and David A Van Leeuwen. Fine-tuning wav2vec2 for speaker recognition. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7967–7971. IEEE, 2022.

[36] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[38] Mayank Sharma. Multi-lingual multi-task speech emotion recognition using wav2vec 2.0. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6907–6911. IEEE, 2022.

[39] Po-Han Chi, Pei-Hung Chung, Tsung-Han Wu, Chun-Cheng Hsieh, Yen-Hao Chen, Shang-Wen Li, and Hung-yi Lee. Audio albert: A lite bert for self-supervised learning of audio representation. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 344–350. IEEE, 2021.

[40] John Hewitt. Designing and interpreting probes, 2019.

[41] Yi-Chen Chen, Shu-wen Yang, Cheng-Kuang Lee, Simon See, and Hung-yi Lee. Speech representation learning through self-supervised pretraining and multi-task finetuning. *arXiv preprint arXiv:2110.09930*, 2021.

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

[43] Raghavendra Pappagari, Jesús Villalba, Piotr Żelasko, Laureano Moro-Velazquez, and Najim Dehak. Copypaste: An augmentation method for speech emotion recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6324–6328. IEEE, 2021.

[44] Wei-Ning Hsu, Anuroop Sriram, Alexei Baevski, Tatiana Likhomanenko, Qiantong Xu, Vineel Pratap, Jacob Kahn, Ann Lee, Ronan Collobert, Gabriel Synnaeve, et al. Robust wav2vec 2.0: Analyzing domain shift in self-supervised pre-training. *arXiv preprint arXiv:2104.01027*, 2021.

[45] Arun Babu, Changhan Wang, Andros Tjandra, Kushal Lakhotia, Qiantong Xu, Naman Goyal, Kritika Singh, Patrick von Platen, Yatharth Saraf, Juan Pino, et al. Xls-r: Self-supervised cross-lingual speech representation learning at scale. *arXiv preprint arXiv:2111.09296*, 2021.

[46] Alexei Baevski, Wei-Ning Hsu, Qiantong Xu, Arun Babu, Jiatao Gu, and Michael Auli. Data2vec: A general framework for self-supervised learning in speech, vision and language. *arXiv preprint arXiv:2202.03555*, 2022.

[47] Rishabh Jain, Mariam Yiwere, Dan Bigioi, and Peter Corcoran. Can self-supervised learning solve the problem of child speech recognition? *arXiv preprint arXiv:2204.05419*, 2022.

[48] Mirco Ravanelli, Jianyuan Zhong, Santiago Pascual, Pawel Swietojanski, Joao Monteiro, Jan Trmal, and Yoshua Bengio. Multi-task self-supervised learning for robust speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6989–6993. IEEE, 2020.

[49] Mirco Ravanelli, Jianyuan Zhong, Santiago Pascual, Pawel Swietojanski, Joao Monteiro, Jan Trmal, and Yoshua Bengio. Multi-task self-supervised learning for robust speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6989–6993. IEEE, 2020.

[50] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[51] Sabrina Stehwien and Ngoc Thang Vu. Prosodic event recognition using convolutional neural networks with context information. *arXiv preprint arXiv:1706.00741*, 2017.

Acknowledgments

This work would not have been possible without the constant support and guidance of my advisor, Prof. Preeti Rao. I am indebted to ma'am for not only her research inputs but also for teaching me the process of conducting systematic research. Looking back, I can clearly see a transformation in my approach towards research and I have Prof. Rao to thank for it. I would also like to express my gratitude towards Kamini Sabu for her valuable inputs throughout this endeavour and experimental results on the baseline work. My sincere thanks to VNS Saketh and Rohit MA, the system administrators of DAP lab, who maintained the GPUs which were used for all experiments in this work. I would also like to thank Nagesh Nayak for his valuable inputs from the perspective of deployment and the demo HTML results. Thanks to Avinash Gaikwad and Asha ma'am for their clerical help. A special shout-out to Hugging Face, the open-source community which hosts the Wav2vec2.0 model we used in this work. As deep learning models grow bigger day-by-day, hosting them in an open-source fashion has become invaluable for those without access to giant clusters of GPUs. I would not have survived this pandemic without my friends, who provided companionship in testing times and made my final year of college memorable. Lastly, this work would have been impossible without my parents, my brother, my grandmother and my extended family who supported me through thick and thin.