Investigating Frame Size Effects on Mental State Classification from the Androids Corpus

Vadim Sokolov

Department of Computer Science, University of Milan, Milan, Italy

***Abstract*—Mental state detection from speech is an important task in clinical cases, that can support early diagnosis and monitoring. In this work, we investigate how varying frame sizes affect the accuracy of mental state prediction using audio from the Androids Corpus. We extract standard acoustic features (MFCCs, deltas, RMS) with varying temporal windows and evaluate performance using linear model, Random Forest and SVM, confirming the hypothesis that mental states vary slowly. We augmented the feature set with additional acoustic descriptors. Results show that nonlinear models achieve higher accuracy than the linear baseline, with the best performance exceeding 80% when using longer frame sizes (5–10 seconds). The added features improved SVM performance in specific configurations but did not benefit all models. These findings highlight the interaction between model choice, feature selection, and temporal resolution in speech based mental state detection.**

I. Introduction

Mental health disorders such as depression affect millions of people worldwide. Automatic detection of such conditions from speech gives a non invasive, scalable, and cost-effective screening mechanism. Speech contains both linguistic and paralinguistic features that can correlate with psychological states.

In this study, our goal is to explore how temporal framing in audio feature extraction affects classification performance. The assumption is that mental states change slowly over time, and thus longer frames might capture more relevant descriptors.

II. Related Work and Motivation

Previous work on the Androids Corpus [1] uses features extracted with OpenSMILE and evaluates classifiers using a speaker-independent 5-fold protocol. Other studies utilized deep learning, but often does not take into consideration the temporal resolution of acoustic features.

Our goal is to explore different frame lengths to understand how temporal granularity affects classification. We hypothesize that longer frames improve performance by capturing more stable features.

III. Methodology

The diagram of the method used is shown in Figure 1. The audio recordings were resampled to 16 kHz and converted to mono. Frame-based feature extraction was performed using window sizes of 20 ms to 30 s, with a 50% overlap. For each frame, the following features were extracted:

MFCCs (13 coefficients) capture the spectral shape of the signal. Delta and Delta-Delta MFCCs (13 each) represent temporal dynamics. Root Mean Square (RMS) energy measures signal power. Fundamental frequency (F0) is estimated using `librosa.yin` [2] for pitch-related information. Harmonic Ratio is calculated as the ratio between harmonic and total energy via Harmonic-Percussive Source Separation (HPSS). Each audio file was converted to a sequence of feature vectors (frames), with an optional aggregation step (mean) or majority voting used later for classification.

For the SVM classifier, an RBF kernel was used. The model was implemented using the Scikit-learn library [3]. Initial hyperparameters were obtained via a combination of grid search and manual tuning: C = 0.3, $\gamma = 0.01$, and class weights 0: 1.0, 1: 0.8. These served as the baseline parameters for experiments prior to adding new features.

A. Dataset

We use the Androids Corpus, which contains recordings from interviews and reading tasks by individuals classified as either healthy controls or patients.

TABLE I: Summary of the Androids Corpus structure and contents.

Component	Description
`Reading-Task/`	112 audio recordings of participants reading a fairy tale. Subfolders: `HC/` (54 files) and `PT/` (58 files).
`Interview-Task/audio/`	116 full interview recordings. Subfolders: `HC/` (52 files) and `PT/` (64 files).
`Interview-Task/audio_clip/`	874 segmented audio clips from interviews, distributed over 116 subdirectories (one per speaker).
`Labels`	Each file is labeled by condition: `PT` (patient) or `C` (control).
`Naming convention`	`nn_XGmm_t.wav`, where fields encode speaker ID, condition, gender, age, and education level.

B. Feature Extraction

We extract the following features using `librosa`:

- 13 MFCCs
- Delta and Delta-Delta of MFCCs
- Root Mean Square (RMS) Energy

We experiment with different frame sizes: 20ms, 30ms, 100ms, 250ms, 500ms, 1000ms, 5000ms, and 10000ms. We study 20s and 30s frame sizes to make sure the patterns no longer persist with drastically wide windows. All features are normalized using standard scaling. The dataset was filtered so no speaker appears in both train and test sets.

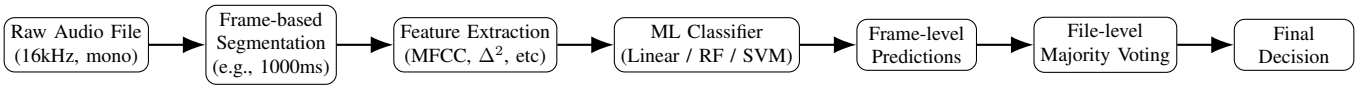


Fig. 1: Overview of the proposed processing pipeline.

C. Classification

We start with a logistic regression model and evaluate frame-level accuracy and F1 score. The dataset is split using a speaker-independent 5-fold division, consistent with the original baseline setup, which can be seen at Table I. Implemented file-level majority voting like a BS2 baseline in the original paper. Compared frame level against file level performance. Different models (Linear, RF, SVM) were tested separately and compared using speaker independent 5-fold cross-validation to perform depression vs. control classification at the frame level.

D. PCA analysis

Principal Component Analysis (PCA) was applied after feature standardization to address the curse of dimensionality. PCA was fitted on the training set and applied to both training and test sets. It retained the minimum number of components to keep at least 95% of the variance. This step was repeated for each frame size and the testing model to provide adaptation to the specific data distribution.

IV. Experiments and Results

A. Evaluation Metrics

For frame level evaluation the following metrics were used:

- Frame-level accuracy
- Frame-level F1 score
- Confusion matrix (TBD)

For file level evaluation the majority voting was used to align with BS2 baseline, improve robustness.

B. Results

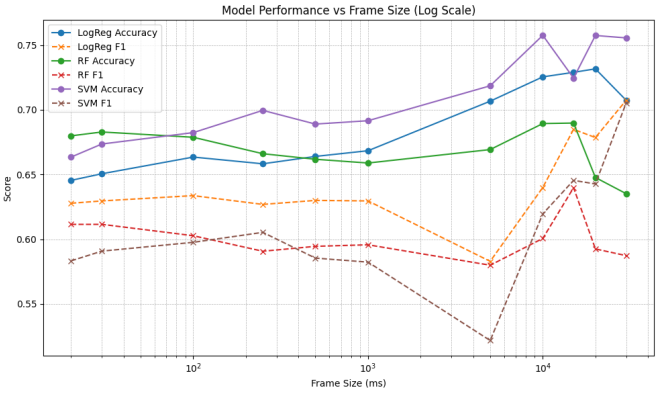


Fig. 2: Accuracy and F1 Score vs Frame Size (log-scale)

Initial results show that performance improves with larger frame sizes, peaking around 10000–20000ms. At the Figure 2 one can see the comparison between different classifiers: Logistic regression, Random Forest, and SVM. Precise results can be analyzed in a Table II, Table III, and Table IV.

C. Feature importance analysis

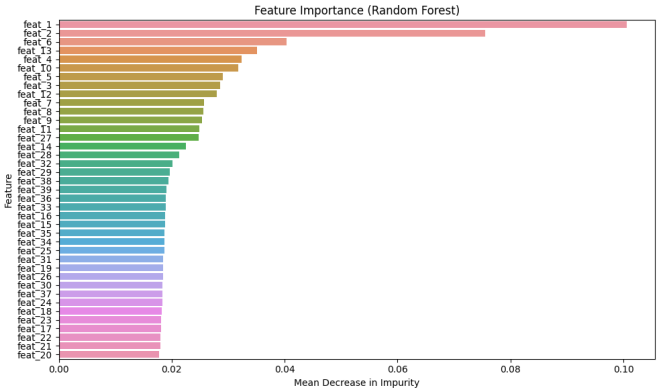


Fig. 3: Feature importances from Random Forest model.

After the performance comparison it was decided to analyze the feature importance and possibly add some more of them to the models. The feature importance analysis was conducted to a Random Forest model. It is shown as an example in Figure 3 for the frame size of 1000ms.

The top features were identified as follows: mmcf_2, mfcc_1, and RMS. These features were not consistent across all frame sizes. The statistics changed for larger frame sizes, where other features like mfcc_13 appeared.

D. Performance of the classifiers with added features

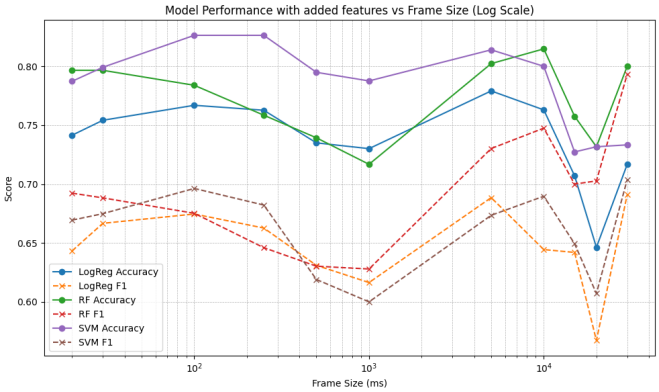
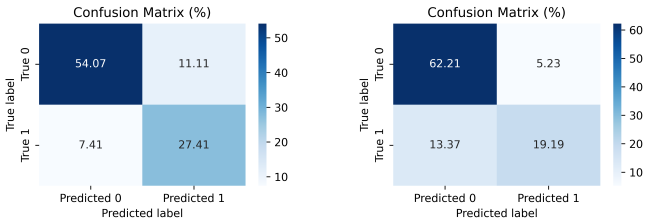


Fig. 4: Accuracy and F1 Score with added features vs Frame Size (log-scale)

We added more features to the models, such as fundamental frequency (F0) and harmonic ratio, to see if they improve

performance Figure 4. The model performance can be seen on Table VI, Table VII, and Table VIII.



(a) Random Forest, frame size = 10s (b) SVM, frame size = 5s

Fig. 5: Confusion matrices (%) for file-level classification.

Figure 5 shows the file-level confusion matrices (in %) for the best-performing configurations of SVM and Random Forest. The model correctly classified 62.21% of class 0 samples and 19.19% of class 1 samples, with most misclassifications coming from class 1 being labeled as class 0 (13.37%). Random forest model achieved a more balanced performance between classes, with a lower false negative rate compared to SVM.

TABLE II: Linear model performance.

Frame size, ms	Accuracy	F1-score
20	0.6455	0.6278
30	0.6506	0.6296
100	0.6635	0.6337
250	0.6583	0.6269
500	0.6640	0.6300
1000	0.6684	0.6296
5000	0.7067	0.5829
10000	0.7255	0.6400
15000	0.7290	0.6849
20000	0.7317	0.6786
30000	0.7074	0.7074

TABLE III: Random forest performance.

Frame size, ms	Accuracy	F1-score
20	0.6799	0.6115
30	0.6829	0.6115
100	0.6788	0.6027
250	0.6661	0.5907
500	0.6618	0.5945
1000	0.6589	0.5957
5000	0.6693	0.5799
10000	0.6894	0.6006
15000	0.6898	0.6398
20000	0.6477	0.5925
30000	0.6352	0.5873

E. PCA analysis

PCA analysis is used to reduce the number of features and therefore address the curse of dimensionality. We are preserving 95% of variance to transform the dataset and get less uncorrelated "principal components" that preserve most of the variance. For example, instead of 42 extended features we got 34-37 features. We used 1000ms, 5000ms, 10000ms, 15000ms frames to study the effect of PCA on accuracy. The tables IX, X, and XI represent classifiers performance with PCA reduced features.

TABLE IV: SVM performance.

Frame size, ms	Accuracy	F1-score
20	0.6635	0.5833
30	0.6736	0.5908
100	0.6824	0.5976
250	0.6996	0.6053
500	0.6890	0.5854
1000	0.6916	0.5823
5000	0.7186	0.5217
10000	0.7576	0.6196
15000	0.7245	0.6455
20000	0.7575	0.6427
30000	0.7556	0.7054

V. Discussion

Larger frames provide better performance, suggesting that mental state-related features are better captured over longer time spans. Short frames likely introduce variability and noise.

A. Feature importance analysis

Directly extracted and visualized feature importances. As a result, one can see that RMS is among the top features. That could indicate energy is a good mental state marker.

TABLE V: Feature importance analysis.

Frame size, ms	Top Feature	2nd Feature	3rd Feature
20	mfcc_2	mfcc_1	rms
30	mfcc_2	mfcc_1	rms
100	mfcc_2	mfcc_1	rms
250	mfcc_2	mfcc_1	rms
500	mfcc_1	mfcc_2	rms
1000	mfcc_1	mfcc_2	rms
5000	mfcc_2	mfcc_1	rms
10000	mfcc_2	mfcc_1	rms
15000	mfcc_1	mfcc_2	rms
20000	mfcc_2	mfcc_1	mfcc_13
30000	mfcc_2	mfcc_1	mfcc_6

From the Table V one can see that the RMS moves from top 3 features to 4th place for the frame size 20s and 30s. When we compute feature importances from a Random Forest using `.feature_importances_`, what we're actually getting is the Mean Decrease in Impurity (MDI). Impurity refers to how mixed the class labels are in a node. Higher MDI equals to a fact that feature was used more often and split more samples while significantly reducing impurity thus, more important.

B. Performance of the classifiers with added features

TABLE VI: Linear model performance with added features.

Frame size, ms	Frame-level Accuracy	Frame-level F1-score	File-level Accuracy	File-level F1-score
20	0.6695	0.6248	0.7415	0.6433
30	0.6718	0.6257	0.7542	0.6667
100	0.6820	0.6318	0.7669	0.6746
250	0.6746	0.6225	0.7627	0.6627
500	0.6767	0.6245	0.7350	0.6310
1000	0.6895	0.6356	0.7301	0.6164
5000	0.7272	0.6756	0.7791	0.6885
10000	0.7274	0.6721	0.7630	0.6444
15000	0.7428	0.7138	0.7071	0.6420
20000	0.6877	0.6489	0.6463	0.5672
30000	0.7133	0.6993	0.7167	0.6909

File-level accuracy was evaluated as well. It was estimated by majority voting as in bs2 baseline. F1 score for random forest is remarkably high in file level estimation, reaching 0.79 fir 30s frames (see Figure 4).

TABLE VII: Random forest performance with added features.

Frame size, ms	Frame-level Accuracy	Frame-level F1-score	File-level Accuracy	File-level F1-score
20	0.6820	0.6290	0.7966	0.6923
30	0.6849	0.6281	0.7966	0.6883
100	0.6813	0.6224	0.7839	0.6752
250	0.6703	0.6130	0.7585	0.6460
500	0.6684	0.6144	0.7393	0.6303
1000	0.6715	0.6272	0.7168	0.6279
5000	0.7437	0.7087	0.8023	0.7302
10000	0.7495	0.7162	0.8148	0.7475
15000	0.7399	0.7239	0.7576	0.7000
20000	0.6957	0.6883	0.7317	0.7027
30000	0.7733	0.7792	0.8000	0.7931

TABLE VIII: SVM performance with added features.

Frame size, ms	Frame-level Accuracy	Frame-level F1-score	File-level Accuracy	File-level F1-score
20	0.6774	0.5848	0.7873	0.6694
30	0.6849	0.6022	0.7992	0.6749
100	0.7169	0.6154	0.8263	0.6963
250	0.7044	0.5912	0.8263	0.6822
500	0.6893	0.5662	0.7949	0.6190
1000	0.6903	0.5646	0.7876	0.6000
5000	0.7465	0.6573	0.8140	0.6735
10000	0.7495	0.6790	0.8000	0.6897
15000	0.7428	0.6942	0.7273	0.6494
20000	0.7589	0.6806	0.7317	0.6071
30000	0.7333	0.7059	0.7333	0.7037

C. PCA analysis

TABLE IX: Linear model performance with PCA-reduced features.

Frame size, ms	PCA	Frame-level Accuracy	Frame-level F1-score	File-level Accuracy	File-level F1-score
1000	no	0.6895	0.6356	0.7301	0.6164
1000	yes	0.6712	0.6131	0.7522	0.6410
5000	no	0.7272	0.6756	0.7791	0.6885
5000	yes	0.7272	0.6767	0.7674	0.6774
10000	no	0.7274	0.6721	0.7630	0.6444
10000	yes	0.7216	0.6734	0.7556	0.6667
15000	no	0.7428	0.7138	0.7071	0.6420
15000	yes	0.7312	0.7138	0.6869	0.6517

PCA for linear classifier gives slightly worse results than without PCA. There are several reasons why PCA hurts linear model performance. First, PCA is unsupervised, it preserves the variance and doesn't preserve the separation of classes. It can remove important discriminative information for the classifier. Second, the used features may already be informative, good in separating classes. PCA might delute their effects.

TABLE X: Random Forest performance with PCA-reduced features.

Frame size, ms	PCA	Frame-level Accuracy	Frame-level F1-score	File-level Accuracy	File-level F1-score
1000	no	0.6715	0.6272	0.7168	0.6279
1000	yes	0.6521	0.6050	0.7434	0.6375
5000	no	0.7437	0.7087	0.8023	0.7302
5000	yes	0.7375	0.6934	0.8256	0.7458
10000	no	0.7495	0.7162	0.8148	0.7475
10000	yes	0.7283	0.6981	0.7704	0.6931
15000	no	0.7399	0.7239	0.7576	0.7000
15000	yes	0.6965	0.6749	0.6869	0.6353

Effect of PCA on File-level accuracy. For linear classifier, PCA improved results only at short frame size of 1000ms. For Random forest, PCA improved the results at shorter durations of 1000ms, 5000ms, possibly reducing overfitting. For SVM, PCA was beneficial at medium and long frames of 5000ms, 15000ms, which is typical for high dimentional models.

To sum up, PCA is not always beneficial. For linear models, PCA removes useful signal unless there is strong redundancy.

TABLE XI: SVM performance with PCA-reduced features.

Frame size, ms	PCA	Frame-level Accuracy	Frame-level F1-score	File-level Accuracy	File-level F1-score
1000	no	0.6903	0.5646	0.7876	0.6000
1000	yes	0.6801	0.5669	0.7522	0.5692
5000	no	0.7465	0.6573	0.8140	0.6735
5000	yes	0.7484	0.6679	0.8256	0.7222
10000	no	0.7495	0.6790	0.8000	0.6897
10000	yes	0.7399	0.6723	0.7704	0.6437
15000	no	0.7428	0.6942	0.7273	0.6494
15000	yes	0.7399	0.7020	0.7374	0.6750

For nonlinear models, PCA helps to reduce the noise and address the curse of dimensionality.

VI. Conclusion

This paper presents an analysis of frame size on speech-based mental state classification. Our experiments show a consistent trend where longer frame sizes (10–20 seconds) yield higher file-level accuracy, supporting the hypothesis that slowly varying speech descriptors carry more discriminative information for this task. We also compared models with and without additional features (fundamental frequency, harmonic ratio) and with PCA-based dimensionality reduction. While these features improved SVM performance in certain frame-size configurations, they did not benefit the linear model and sometimes decreased performance due to the curse of dimensionality. PCA occasionally improved results, particularly for Random Forest and SVM with smaller frame sizes (1–5 seconds), but generally led to lower frame-level accuracy. The best file-level accuracy achieved was above 80% using Random Forest (10s frames) and SVM (5s frames), with majority voting per file as in the BS2 baseline from the original dataset paper. These findings confirm that longer analysis windows help capture stable characteristics relevant to mental state detection while model choice and feature selection interact strongly with the optimal frame size.

References

[1] Alessandro Vinciarelli, University of Glasgow et al. *The Androids Corpus: A New Publicly Available Benchmark for Speech Based Depression Detection*. Interspeech 2023.

[2] Brian McFee et al. *librosa: Audio and music signal analysis in Python*. Proceedings of the 14th python in science conference. 2015.

[3] Pedregosa et al. *Scikit-learn: Machine Learning in Python*. JMLR 2011.