Ensemble Model for Detecting Infant Cries, Screams, and Normal Utterances

Pinakin Choudhary
BTech in Mathematics and Computing
Indian Institute of Science
Bengaluru, India
pinakinc@iisc.ac.in

***Abstract*—I have trained YAMNet and Wav2Vec2 architecture based models and used ensemble of these to develop a robust audio classification system capable of distinguishing between infant cries, screams, and normal utterances. This report contains the exact details of implementation, workflow and explanation behind design choices such as Datasets, Data Augmentation techniques, Model Selection, Ensemble techniques and Loss function. It also includes all the results of performance and evaluation metrics such as ROC curve, confusion matrices, and classification reports.**

I. Data Aquisition and Preprocessing

This section contains Dataset choices, Data Augmentation techniques, and Preprocessing steps.

A. Datasets

I have used the following datasets for training the models:

- **Infant's Cry Sound:** This dataset contains 1000 audio files of infant cries, screams, and normal utterances. The dataset is available on Mendeley Data [**?**].
- **Human Screaming Detection Dataset:** This dataset contains 1000 audio files of human screams. The dataset is available on Kaggle [**?**].
- **Child Speech, Kid Speaking:** This dataset contains 1000 audio files of child speech and kid speaking. The dataset is available on AudioSet [**?**].

B. Data Cleaning

Audioset had lots of broken links and missing files (downloaded using `audioset-downloaded`). Had to manually prepare the dataset by collecting different files from all three datasets. Used `script.ipynb` to spilt everything into 5 sec clips. Removed excess and damaged files.

The directory structure for the datasets is as follows:

```
data/
+-- cry/
|   +-- 132 .wav files 5 sec each
+-- speech/
|   +-- 132 .wav files 5 sec each
+-- scream/
|   +-- 132 .wav files 5 sec each
```

Note: Strictly YAMNet and Wav2Vec2 can handle variable length audio files. But for uniformity in training, I have used 5 sec clips for all models. This is not the case in inference, where variable length audio files are supported.

Each subdirectory contains 132 .wav files. And I used 80:20:10 train:val:test split. So about 8 minutes of train audio, 2 minutes of val audio, and 1 minute of test audio for each category.

C. Data Augmentation

I referred to the paper [**?**] for audio data augmentation techniques. Following data augmentaion was performed to make model more robust and prevent overfitting:

- Normalization: Amplitude normalization
- Time Stretching: Applied with a random rate between 0.8x and 1.2x to simulate variations in speaking speed, enhancing model robustness.
- Time Shifting: Random shift up to 0.5 sec to make the model invariant to temporal shifts, improving its ability to handle different starting points.
- Pitch Shifting: Random pitch shift of ±2 semitones to account for variations in pitch, making the model more resilient to different voice tones.
- Background Noise Addition: Random noise at 5-10% of signal amplitude to simulate real-world noisy environments, improving the model's noise robustness.
- Dynamic Range Compression (DRC): Applied to enhance percussive elements, ensuring the model can better detect transient sounds.

D. Preprocessing

The audio files were converted to 16kHz mono channel audio files. And converted to tf tensors and pytorch tensors respectively for YAMNet and Wav2Vec2 models. Train, eval, test split applied.

II. Model Selection

This section contains the details of the models used for training and their architecture.

A. YAMNet

YAMNet [**?**] is a pretrained deep net that predicts 521 audio event classes based on the AudioSet-YouTube corpus, and employing the `Mobilenet_v1` depthwise-separable convolution architecture.

I have used the 1024-dimensional embedding layer output of YAMNet as input to a custom Dense Multi-Layer Perceptron (MLP) classifier with final output layer with 3 neurons for the 3 classes. The architecture of model is as follows:

TABLE I
YAMNET-BASED MLP CLASSIFIER ARCHITECTURE

Layer (type)	Output Shape	Param #
Dense (ReLU)	(None, 256)	262400
Dropout (0.3)	(None, 256)	0
Dense (ReLU)	(None, 128)	32896
Dropout (0.3)	(None, 128)	0
Dense (Softmax)	(None, 3)	387

Model Training and Evaluation: The model was compiled with the Adam optimizer, a learning rate of 0.001, and categorical cross-entropy loss. Accuracy was used as the evaluation metric.

Class weights were computed to handle class imbalance using the `compute_class_weight` function from `sklearn`. Although this was redundant because I had already taken balanced dataset during cleaning. But it's still a good practice to handle class imbalance which may be introduced while dataset splits, etc.

The model was trained for 30 epochs with a batch size of 32, using the computed class weights.

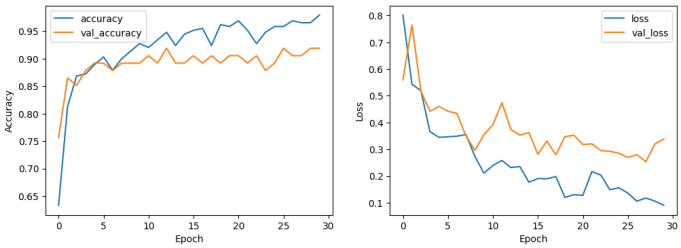

Fig. 1. Training and Evaluation Loss Curves

Evaluation on Test data: The model was evaluated on the test data, and the following results were obtained:

TABLE II
CLASSIFICATION REPORT YAMNET

Class	Precision	Recall	F1-Score
Cry	1.00	1.00	1.00
Scream	1.00	0.82	0.90
Speech	0.85	1.00	0.92
Accuracy		0.94	
Macro Avg	0.95	0.94	0.94
Weighted Avg	0.95	0.94	0.94

TABLE III
CONFUSION MATRIX YAMNET

	Cry	Scream	Speech
Cry	11	0	0
Scream	0	9	2
Speech	0	0	11

B. *Wav2Vec2*

Wav2Vec2 [**?**] is a pretrained model for speech recognition. It also can be used as feature extractor with 768-dimensional embeddings. I have tried two approaches:

- Using the pretrained model as a feature extractor and training a custom MLP classifier on top of it.
- Fine-tuning the pretrained model on the dataset.

Model Training and Evaluation: I have used the first approach as it gave better results. Fine-tuning the model took a long time and even after that results were not satisfactory. Maybe more epochs would be required, but that comes at the cost of important time and compute resources. The extracted features were passed through a custom MLP classifier with following architecture:

TABLE IV
WAV2VEC2-BASED MLP CLASSIFIER ARCHITECTURE

Layer (type)	Output Shape	Param #
Dense (ReLU)	(None, 256)	262400
Dense (Softmax)	(None, 3)	771

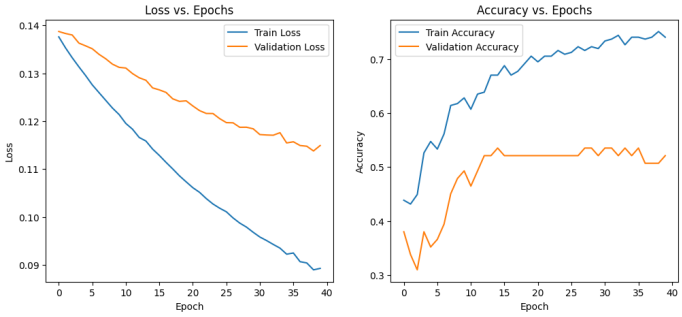

Fig. 2. Training and Evaluation Loss Curves for Wav2Vec2-based MLP Classifier

Evaluation on Test data: The model was evaluated on the test data, and the following results were obtained:

TABLE V
CLASSIFICATION REPORT WAV2VEC2

Class	Precision	Recall	F1-Score
Class 1	1.00	0.90	0.95
Class 2	0.43	0.38	0.40
Class 3	0.50	0.64	0.56
Accuracy		0.725	
Macro Avg	0.64	0.64	0.64

TABLE VI
CONFUSION MATRIX WAV2VEC2

	Cry	Scream	Speech
Cry	19	0	0
Scream	0	3	4
Speech	2	5	7

III. Ensemble Model

An ensemble of YAMNet and Wav2Vec2 models was utilized to enhance robustness and improve overall performance. Three different ensemble techniques were explored:

A. Ensemble Techniques

I explored multiple ensemble learning techniques to enhance classification performance. Ensemble methods reduce variance and improve generalization. The following ensemble techniques were implemented and evaluated:

1) Average: In this technique, the probabilities predicted by each model are averaged to get the final prediction. This helps in smoothing out the predictions and reducing the variance.

2) Weighted Average: In this technique, the probabilities predicted by each model are weighted according to their performance on the validation set. The final prediction is obtained by averaging these weighted probabilities. This helps in giving more importance to the better performing model.

3) Hard Voting: In this technique, the final prediction is obtained by taking the majority vote of the predictions made by each model. This helps in reducing the bias and making the final prediction more robust.

Evaluation on Test data: The ensemble model was evaluated on the test data, and the following results were obtained:

TABLE VII
Classification Report Ensemble Model (Simple Average)

Class	Precision	Recall	F1-Score
Cry	0.92	1.00	0.96
Scream	1.00	0.92	0.96
Speech	1.00	1.00	1.00
Accuracy		0.97	
Macro Avg	0.97	0.97	0.97
Weighted Avg	0.97	0.97	0.97

TABLE VIII
Confusion Matrix Ensemble Model (Simple Average)

	Cry	Scream	Speech
Cry	12	0	0
Scream	1	11	0
Speech	0	0	12

TABLE IX
Classification Report Ensemble Model (Hard Voting)

Class	Precision	Recall	F1-Score
Cry	0.75	1.00	0.86
Scream	0.69	0.75	0.72
Speech	1.00	0.58	0.74
Accuracy		0.78	
Macro Avg	0.81	0.78	0.77
Weighted Avg	0.81	0.78	0.77

TABLE X
Confusion Matrix Ensemble Model (Hard Voting)

	Cry	Scream	Speech
Cry	12	0	0
Scream	3	9	0
Speech	1	4	7

TABLE XI
Classification Report Ensemble Model (Weighted Average)

Class	Precision	Recall	F1-Score
Cry	0.92	1.00	0.96
Scream	1.00	0.92	0.96
Speech	1.00	1.00	1.00
Accuracy		0.97	
Macro Avg	0.97	0.97	0.97
Weighted Avg	0.97	0.97	0.97

B. Loss Function

The loss function used for training both the models was categorical cross-entropy. This is a standard loss function for multi-class classification problems. It calculates the cross-entropy loss between the predicted probabilities and the true labels.

$$L(y, \hat{y}) = -\sum_{i} y_i \log(\hat{y}_i) \tag{1}$$

where y is the true label, $\hat{y}$ is the predicted probability, and i is the class index.

While training the model with best validation accuracy was saved. And later these weights were used to evaluate the model on test data.

IV. Deployment with Temporal

The ensemble model was deployed using Temporal and Streamlit for UI.

The Streamlit UI has two sections:

- **Upload Audio:** Users can upload an audio file to classify it as either a cry, scream, or speech.
- **Record Audio:** Users can record audio using their microphone and classify it as either a cry, scream, or speech.

The Temporal workflow consists of the following steps:

- **Preprocess Audio:** If the audio is recorded, it is first saved as a .wav file. Now, the saved .wav files are preprocessed to convert them to 16kHz mono channel audio files. This allows real time audio processing, User recorded audio is saved and preprocessed.

TABLE XII
Confusion Matrix Ensemble Model (Weighted Average)

	Cry	Scream	Speech
Cry	12	0	0
Scream	1	11	0
Speech	0	0	12

- **Classification** The preprocessed audio files are then passed through the ensemble model to classify them as either a cry, scream, or speech.
- **Display Results:** The classification results with confidence are displayed to the user in the Streamlit UI.
- **Save Results:** The classification results are saved in a .json file for future reference.

As mentioned above, during inference the extraction of both model doens't require fixed length audio files. So, given any audio file, the features are extracted and passes to respective MLP classifier. Which in turn is used by ensemble to predict the label with a confidence score.

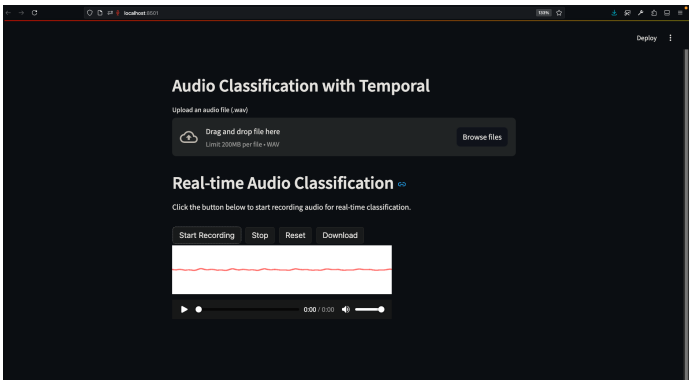

Fig. 3. Streamlit UI for Audio Classification

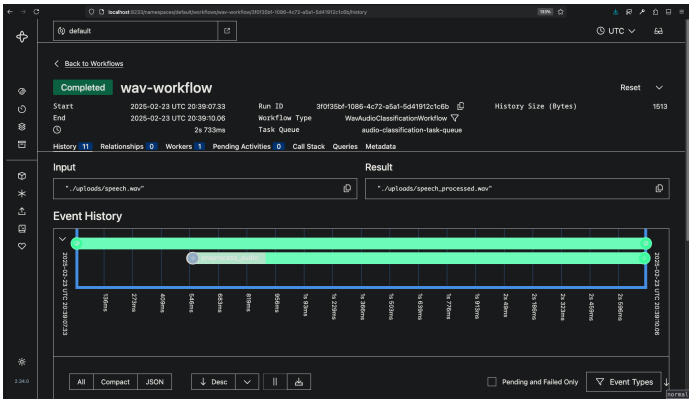

Fig. 4. Temporal UI for Workflow Management

REFERENCES

[1] Rosita, Yesy Diah (2020), "Infant's Cry Sound", Mendeley Data, V1, doi: 10.17632/hbppd883sd.1

[2] Kaggle: Human Screaming Detection Dataset, https://www.kaggle.com/datasets/whats2000/human-screaming-detection-dataset.

[3] AudioSet: Child speech, kid speaking, https://research.google.com/audioset/dataset/child_speech_kid_speaking.html.

[4] J. Salamon and J. P. Bello, "Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification," in IEEE Signal Processing Letters, vol. 24, no. 3, pp. 279-283, March 2017, doi: 10.1109/LSP.2017.2657381.

[5] YAMNet model, https://tfhub.dev/google/yamnet/1.

[6] Baevski, Alexei, et al. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations." arXiv preprint arXiv:2006.11477 (2020). https://arxiv.org/abs/2006.11477.