Pattern Recognition Final Report
Respiratory Sound Classification: Voting with Transformer Embeddings

111061702 An-Yan Chang (常安彥)

I. Problem Description

1. Motivation

   Respiratory sounds with different features may indicate various diseases:

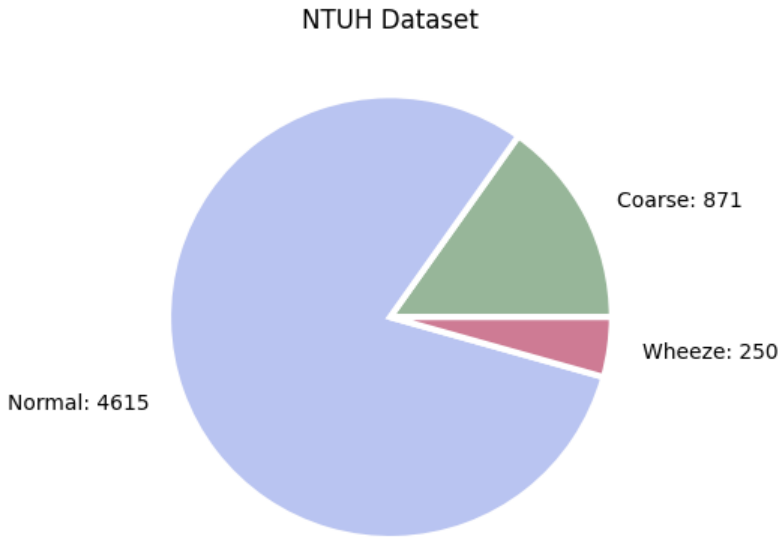
   a. Coarse crackle: A thick respiratory sound, similar to bubbles. These sounds occur if fluid fills the small air sacs in lungs and air moves in them. The air sacs fill with fluid when a person has pneumonia or heart failure.

   b. Wheezing: A high-pitched thin sound, like whistle. This sound occurs when the bronchial tubes become inflamed and narrowed.

   By the model's diagnosis, it is hoped that doctors can effectively screen the potential causes of a patient's condition, or identify patterns that may have been overlooked by human observation. Furthermore, hope that the model can replace more painful testing methods, such as rapid screening.

2. Dataset

   Formosa archive of breath sound: The dataset is collected in collaboration with NTUH, and includes respiratory sounds from patients. The sounds range from coarse to wheezing and normal healthy breaths, with a **highly unbalanced distribution** where normal breaths constitute the majority. Furthermore, this dataset is currently not publicly available, and will be released to the world later this year, focusing on respiratory sounds of Asian individuals.



[fig1.] Data distribution of *Formosa archive of breath sound* (Normal: 4615, Coarse: 871, Wheeze: 250)

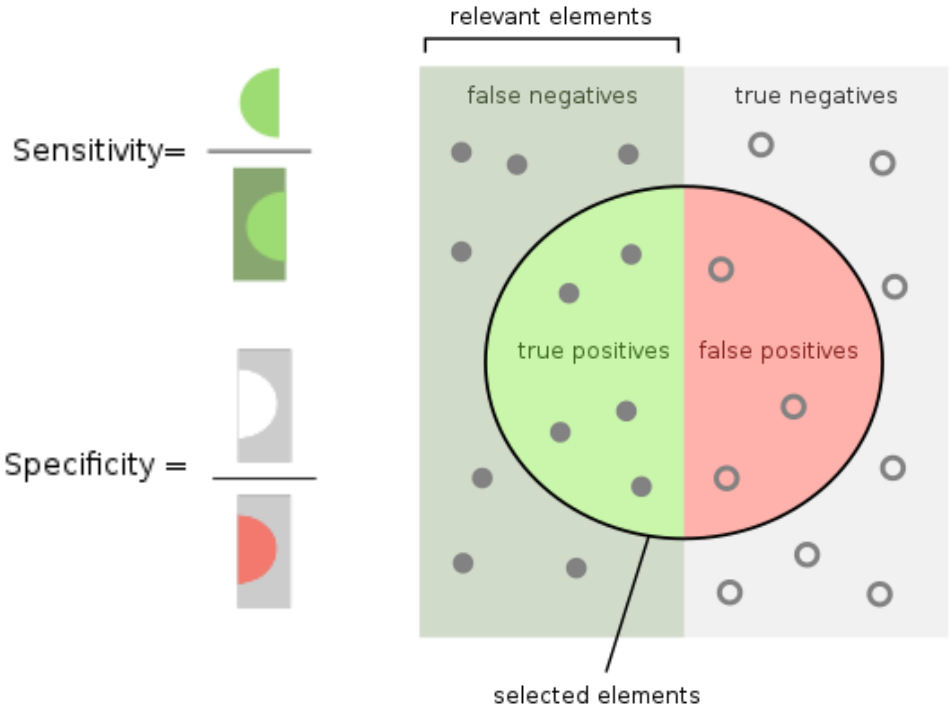
II. Methods and Experiments

1. Evaluation methods: ICBHI score

Respiratory cases mostly evaluate the performance with ICBHI score, this is created by the team of ICBHI dataset[1], the biggest eastern respiratory sound dataset. In the following formula, it is obvious that the score is equally distributed of sensitivity and specificity, which means the performances of both normal and abnormal respiratory sounds are important.

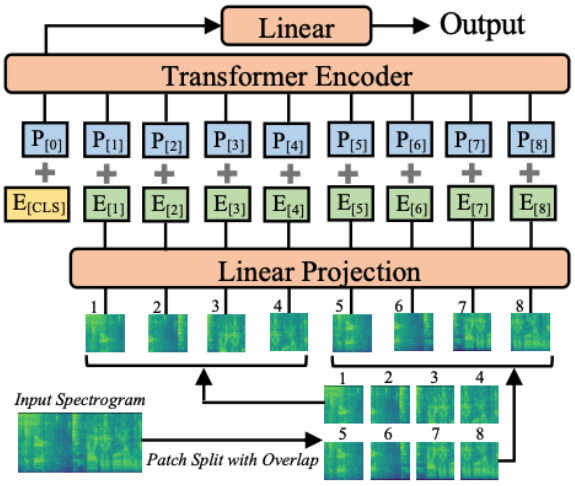
ICBHI score = (Sensitivity + Specificity) / 2

(Positive includes ‘coarse’ and ‘wheeze’; negative includes ‘normal.’)



[fig2.] visualization of sensitivity and specificity, which are recall of normal and abnormal

2. Embedding Features: Audio spectrogram transformer[2]



[fig3.] overall structure of audio spectrogram transformer from the original paper, AST[2]

In order to extract the embedding features of these audio files for training, I built an audio spectrogram transformer. The structure of the transformer refers to AST[2], which splits the spectrogram into patches, combining position embeddings, and train as an encoder.

a. Pretrain stage

Since the number of instances of *Formosa archive of breath sound* was few, I pretrained a transformer with large public dataset to learn how to encode audio files. AudioSet[3], including a million audio files from YouTube, was the chosen dataset.

Due to lack of labels in AudioSet, I included self-supervised learning in this pretrain stage to not discard huge amount of data. Noise-contrastive estimation loss was used for cluster training, and it acted like binary MLE loss after sigmoid function.

b. Finetune stage

See my respiratory sound dataset as a downstream task, and train on it after loading the model state dictionary of my previous pretrained model without freezing base.

I got the embedding features from the last layer of my transformer, and took these features as the input of my following methods. The features are 1536-dimensional, which was very high, and thus, feature extraction was my next step.

3. Feature Extraction

Standardize the features to avoid the diverse range of the values, and then implement the following methods once at a time.

a. No extraction

Some methods were not suitable for feature extraction or already included dimension reduction in their algorithms.

b. PCA

n_components = 0.8

n_components = 0.2

c. LDA

shrinkage = 0.2

shrinkage = 0.8

d. Decision tree classifier

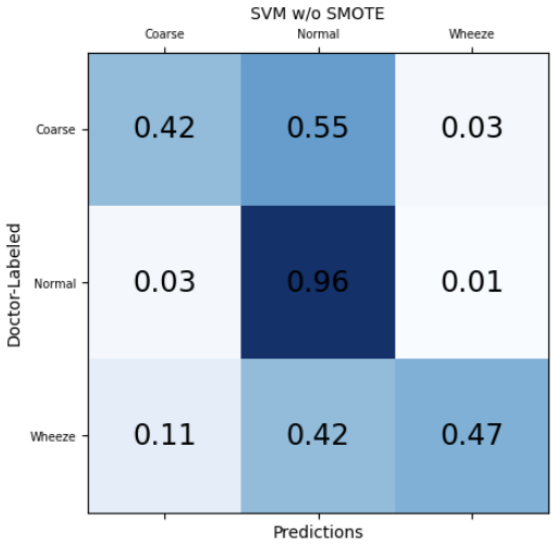
10 most important features

50 most important features

4. SVM

First, I implemented SVM as a classifier to verify the feasibility of the methods above, because SVM is a fast and well perform classifier.

- acc: 0.856
- spe:0.962
- sen: 0.43
- score: 0.696



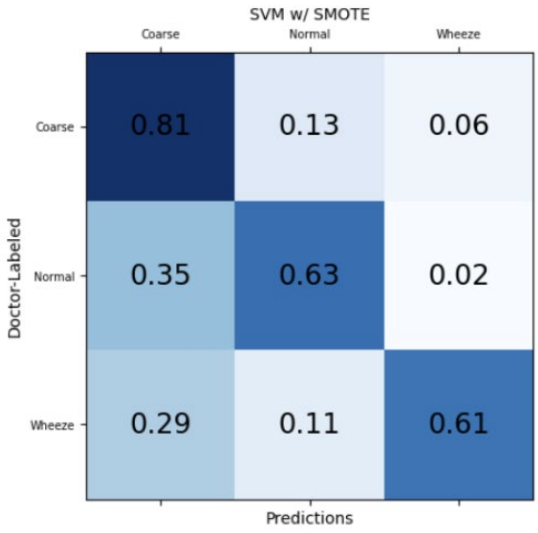
As the result shown above, although accuracy was high and specificity was nearly perfect, sensitivity performs badly. This was not a result we want, and it was clearly caused by data imbalance of our dataset. On the other hand, the result showed that my embedding features worked on this case.

5. SMOTE: deal with data imbalance

SMOTE aims at dealing with data imbalance issues with oversampling algorithm. It generates new samples between exist samples of minor labels to supplement the lack of instances. Details as below:

$$x_{generated} = x_{chosen} + (x_{nearest} - x_{chosen}) * \delta$$

- acc: 0.657
- spe:0.627
- sen: 0.775
- score: 0.701

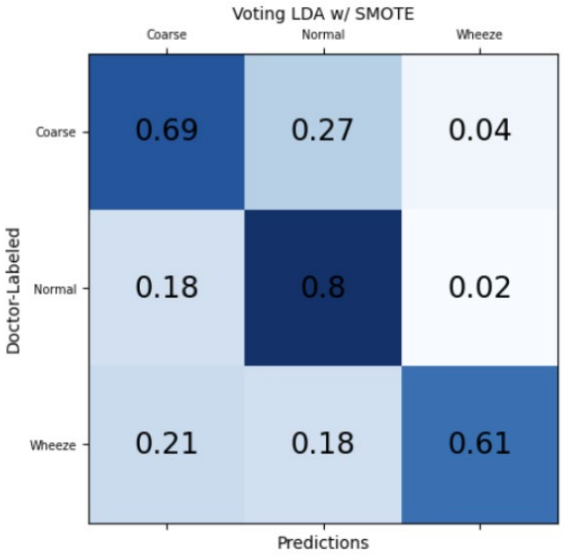


As the result shown above, although specificity became lower, sensitivity and ICBHI score significantly improved. This balanced performance was a good result that I wanted. Therefore, my next step is pushing performance.

6. Voting Ensemble: pushing performance

I ensembled 5 models to vote for the prediction, including SVM, KNN, random forest, XGBoost and Lightgbm. Expect the methods that the course covers, XGBoost and LGBM are both gradient boosting algorithms, where LGBM is lighter, generating leaves per iteration.

- acc: 0.776
- spe: 0.801
- sen: 0.675
- score: 0.738

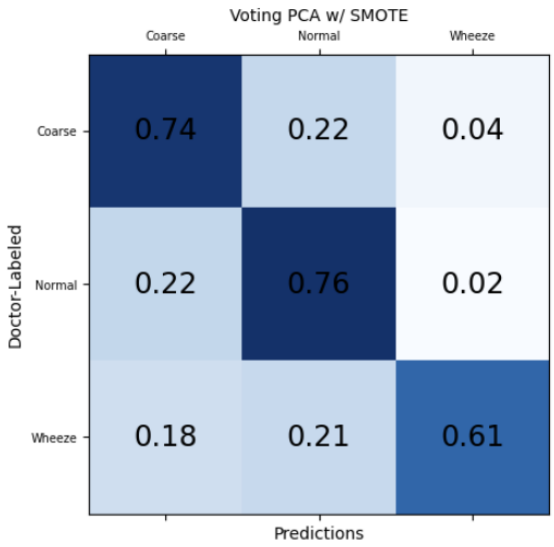


This ensemble model had a better result with the highest score, 0.738. Nevertheless, the sensitivity should be able to hit a better score. I assumed that it was caused by lack of abnormal dataset, unable to generalize.

7. Augmentation: generalize

I implemented 5 augmentation methods, where 4 of them refers to this paper[4], including mixup, loudness, audio speed, and frame shifting. The fifth method was masking the patches of spectrogram, training input of transformer.

- acc: 0.748
- spe:0.756
- sen: 0.715
- score: 0.736



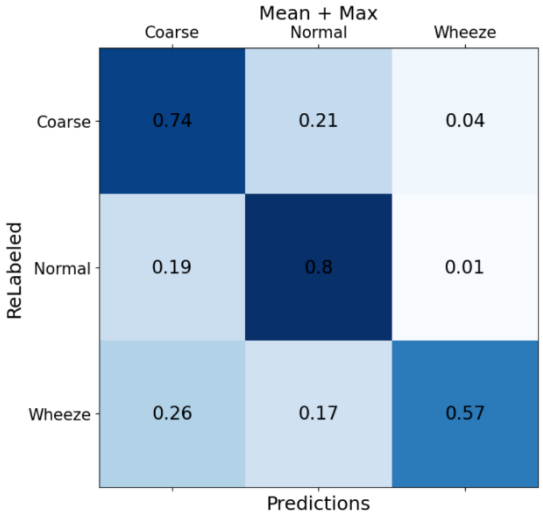
The sensitivity increased as I expected, and the overall score remained at the same level. This was an ideal performance, but I would also like to train on MLP methods.

8. MLP methods

When finetuning audio classification tasks with MLP in downstream of transformer, 'mean pooling' is often implemented to extract the global features of the whole audio, such as speaker or emotion. Nevertheless, respiratory classification is not the same. According to this paper[5], the sound of coarse is discontinuous, and wheeze's is continuous.

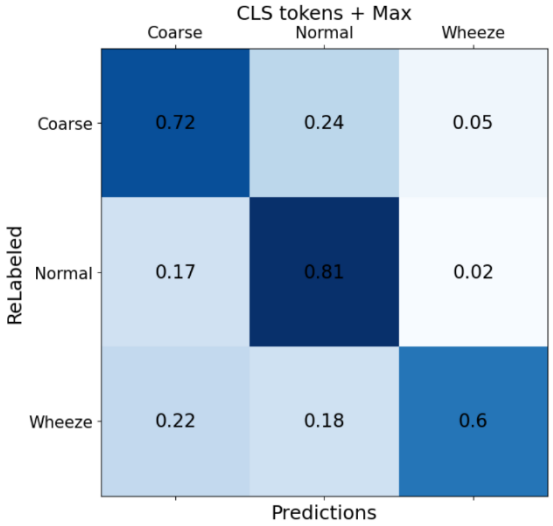
a. Mean pooling + Max pooling: I used mean pooling to capture the overall feature, such as continuous sound like wheeze, and used max pooling to capture prominent voice, such as coarse.

- acc: 0.782
- sensitivity: 0.707
- specificity: 0.799
- score: 0.753



b. CLS token[6] (+ Max pooling): I used CLS token, which also aims to capture overall signal, instead of mean pooling.

- acc: 0.789
- sensitivity: 0.691
- specificity: 0.812
- score: 0.751



Both of them gave a better score, and hit the highest on ICBHI score so far. With both pooling methods, I achieved 0.7 on both specificity and sensitivity.

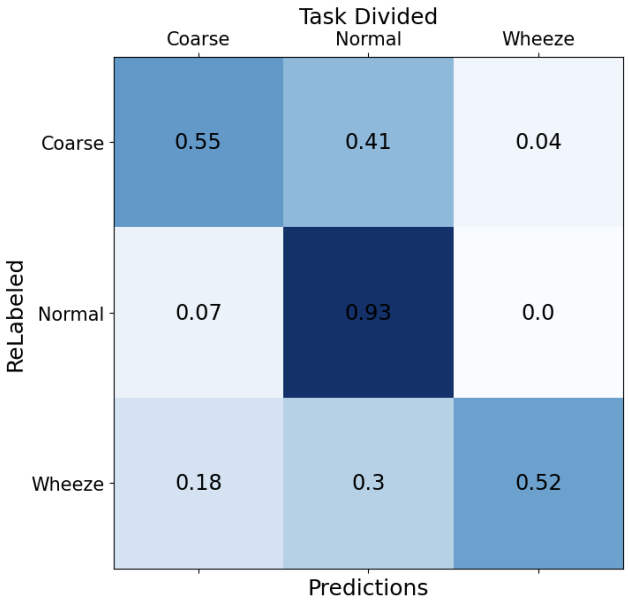
9. Divide and Conquer:

Coarse-	precision	Recall	F1	Wheeze	precision	Recall	F1
Voting	0.415	0.724	0.529	Voting	**0.607**	**0.557**	**0.577**
MLP	**0.419**	**0.799**	**0.534**	MLP	0.576	0.495	0.533

As the summary of the performances of both methods shown above, voting classifier behaved better on wheeze, and MLP had a better score on coarse. Therefore, I divided the task to include both advantages.

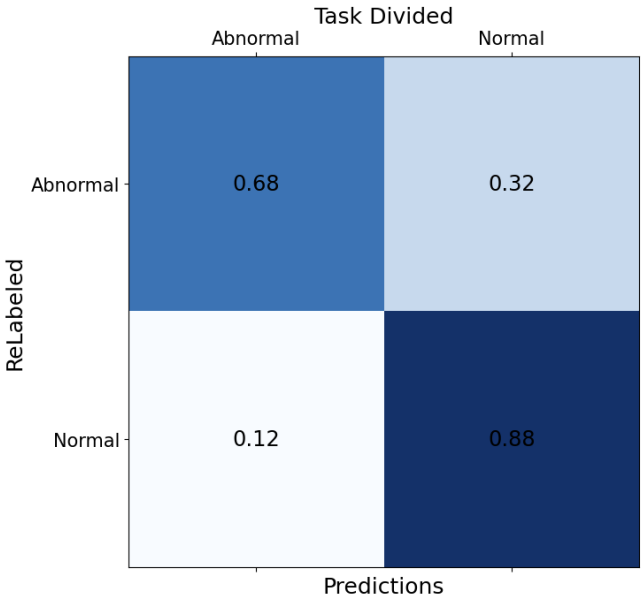
a. (Coarse, Normal), Wheeze: Voting classifier split 'wheeze' out, then, MLP separated coarse and normal.

- acc: 0.861
- sensitivity: 0.540
- specificity: 0.931
- score: 0.736



As the result shown above, sensitivity was really low. I assumed that the class wheeze had too few samples compared to (coarse + normal).

b. Normal, (Coarse, Wheeze): Voting classifier split 'normal out, then, MLP separated coarse and wheeze.



In the first step, the model already performed badly. 1/3 of the abnormal samples were defined incorrectly. It seemed that dividing tasks made the imbalance more significant.

III. Conclusion and Comparison

1. Summary

Methods	ACC	spe	sen	score
SVM	0.856	0.962	0.43	0.696
SVM w/ SMOTE	0.657	0.627	0.775	0.701
Voting w/ SMOTE w/o extraction	0.644	0.609	0.785	0.697
Voting w/ SMOTE w/ LDA	0.776	0.801	0.675	0.738
Voting w/ SMOTE w/ PCA w/ AUG	0.748	0.756	0.715	0.736
MLP mean + max	0.782	0.707	0.799	0.753
MLP CLS + max	0.789	0.691	0.812	0.751
(Coarse, Normal), Wheeze	0.861	0.931	0.540	0.736

2. Comparison with SOTA

To make comparison with the state-of-the-art, I had to evaluate my methods on the largest public dataset, ICBHI[1]. The dataset is also imbalanced with four labels, both refers to both coarse and wheeze sound.

Methods	ACC	spe	sen	score
My Voting	0.725	0.771	0.737	0.754
My MLP	0.797	0.853	0.733	0.793
Glance-and-Gaze	0.847	0.845	0.849	0.847
Snapshot ensemble	--	0.861	0.691	0.776
CRNN	0.816	0.777	0.855	0.816
BiGRU + ATT	0.766	0.773	0.760	0.766

My methods scored around 0.75 to 0.79, when the best model scored 0.847 and the second model scored 0.816. Compared to the three SOTA below, my models were competitive, and showed that these methods were feasible.

IV. References

[1] https://bhichallenge.med.auth.gr/ICBHI_2017_Challenge

[2] (arXiv:2104.01778) AST: Audio Spectrogram Transformer, Yuan Gong, Yu-An Chung, James Glass

[3] https://research.google.com/audioset/download.html

[4] Deep learning based respiratory sound analysis for detection of chronic obstructive pulmonary disease, PeerJ 2021, Arpan Srivastava, Sonakshi Jain, Ryan Miranda, Shruti Patil, Sharnil Pandya, Ketan Kotecha

[5] Respiratory sound classification for crackles, wheezes, and rhonchi in the clinical field using deep learning, 2021, Yoonjoo Kim1,5, YunKyong Hyon2,5, Sung Soo Jung1, Sunju Lee2, Geon Yoo3, Chaeuk Chung1,4* & Taeyoung Ha2

[6] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018, Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova