

ECE 228 PROJECT: COVID-19 COUGHING SOUND PREDICTION

Group 34: Jiajun Du, Kunpeng Liu, and Haoming Zhang

*University of California San Diego, La Jolla, CA 92093-0238

ABSTRACT

The existing methods for detecting COVID-19 are mostly physical, which means that a physical contact between patients and doctors is needed. Such contact would increase the chance of infection for doctors. Thus, the purpose of this project is to find a non-contact way of detecting COVID-19. Our method is to use the coughing sound dataset to train multiple machine learning models to detect COVID-19 by only collecting the coughing sound of a patient that is potentially infected. It turns out that coughing sound and COVID-19 has strong relationship. The models we used are: Convolutional Neural Network, Simplified Neural Network, Random Forest, Boosting and Auto Encoder. The overall accuracy of all the models are over 95%.

Index Terms— Convolutional Neural Network, Simplified Neural Network, Random Forest, Boosting, Auto Encoder

1. INTRODUCTION

1.1. Background

COVID-19 is one of the most serious epidemics in human history and it has now spread over the whole world. By this time, there are totally over 7 million people[1] were detected positive to COVID-19 and there are more than 400 thousand people[1] were killed by this epidemic. There are several ways to detect if one has been infected, one of them is Polymerase chain reaction[2], a way to physically detect by collecting patients' nasopharyngeal swab or sputum sample. Among all testing techniques, they need physical contact between patients and doctors, which have some potential infection danger. For this project, we will use the audio(coughing) data to build a model that can give patients detection results by only recording the sound of coughing and sending it to our model. This method will reduce the danger of potential infection issues. We implemented the Short Time Fourier Transform(STFT) to represent the spectrum of frequencies of audio as they vary with time, then extracted some features like zero crossing rate or unnamed features extracted using deep learning. For feature extraction, we referred to Music Feature Extraction in Python[3]. Then, we classified them into "positive" or "negative" based on these features, which

referred to Classification of Music into different Genres using Keras[4]. After that, we implemented our deep learning models and trained with our coughing sound dataset.

1.2. Input and Output format

The input to our algorithm was .wav sound files. We then used multiple machine learning models (Convolutional Neural Network, Simplified Neural Network, Random Forest, Boosting and Auto Encoder) to train our dataset and to find the relationship between coughing sound and COVID-19.

2. RELATED WORK

2.1. AI4COVID-19: AI Enabled Preliminary Diagnosis for COVID-19 from Cough Samples via an App[5]

The design for AI4COVID-19 was to build application that allowed user to record coughing sound and detect through the model they built and output the diagnose of positive or negative of COVID-19. We did not implement the front-end as the same as this project but the inner machine learning blocks containing Deep Learning-based Multi-Class classifier (DL-MC), Classical Machine Learning-based Multi-Class classifier (CML-MC), and Deep Learning-based Binary-Class classifier (DL-BC) gave us some intuition of what we can progress to.

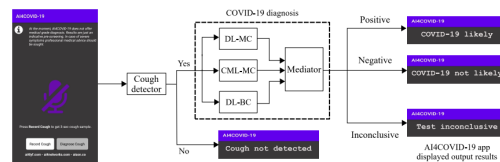


Fig. 1. Workflow for AI4COVID-19[5].

As figure 2 from Ref. [5] shows, the design was a multi-layer convolutional neural network that the group designed to train their model and according to the output, the accuracy could be more than 90%. Based on this design, we developed our CNN model and besides that.

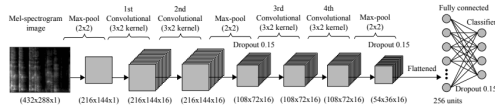


Fig. 2. CNN Layer design for AI4COVID-19[5].

2.2. Classification of Music into different Genres using Keras[4]

The article introduced how each feature of music can be extracted and what role each feature takes place for the whole music file. It also gave a trivial example of how to use those features to develop a neural network to classify the genre of music. The topic was not exactly the same we are working on, but the method of extracting features and building models enlightened us a lot.

3. DATASET AND FEATURE

We used the dataset on Canvas website, with 1288 positive cough data files and 90 negative cough data files. All the original files were in ".wav" type. Then we transformed the original data into short-time fourier transform spectrogram, which can help us to extract different features easier. Here we split the training data and testing data as rate 4:1. Below is an example of the original data (figure 2) and its short-time fourier transform spectrum(figure 3). After we got the short time fourier transform, we can get these 6 types of features: 1. Chromagram. This features can capture the harmonic and melodic characteristics of a audio file 2. The zero-crossing rate is a key feature in the speech recognition and music processing. Briefly speaking, it shows the rate of changing, which means when the signal goes across zero, it will capture this kind of change and record it. 3. Spectral Centroid. This feature can help us to located where the center of a spectrum. Usually, the center of the spectrum is a very significant feature that can help us locate where is the "middle" of the spectrum. 4. Spectrum Bandwidth. This feature is basically showing the difference between the lowest frequency in a spectrum and the highest frequency in a spectrum. 5. Spectral Rolloff. This feature can tell us, under which frequency, we can get how much percentage of the total energy for the signal. 6. Mel Frequency Cepstral Coefficients (MFCCs). It is a set of several small features that included in the spectrum. Basically, This feature is a very powerful that can be used in any audio processing such as speech recognition and music classification.

4. METHODS

In this section, we will describe all models used in our experiments, CNN, Autoencoder, Random Forest and Boosting. We are trying to use these different models to give a convincing result.

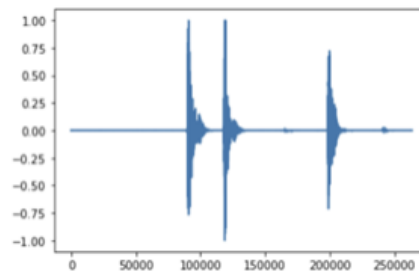


Fig. 3. Original data.

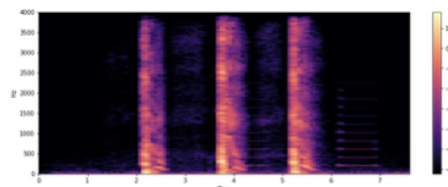


Fig. 4. Short time fourier transform.

4.1. CNN

Convolutional Neural Networks(CNNs) are widely implemented in a variety of machine learning applications, such as image classification, face recognition. The CNN algorithm is constructed by stacking multiple computation layers for feature extraction and classification [6]. A sequential layers of CNNs transform the input data into highly abstract representations called feature maps and filters are applied in the feature maps to extract embedded characteristics and generate the output[7].

Especially in image classification, convolutional layers are able to capture the Spatial and Temporal dependencies, so it succeeds in the field of image recognition. As Figure 5 shows, one of the examples for typical application of CNN is handwriting classification. Although the objects we need to classify is audio here, after feature extraction, we can get 1-D data which is an important step for all experiments in this project and then convert it into 2-D, which can be processed by convolutional layers like images.

4.2. Random Forest

The random forest algorithm is a very popular machine learning algorithm based on the decision trees. Decision tree is a method to make decisions based on conditions and statements. It consists of three types of nodes: decision nodes, chance nodes and end nodes. It will make decisions under particular rules. Random fores is a combination of decision tress, which can combine multiple decision trees into one decision using "vote" algorithm. Good reasons for using random forest algorithm: 1. highly accurate and robust because of the number of decision trees participating in the process. 2. It can be used both in regression and classification process.

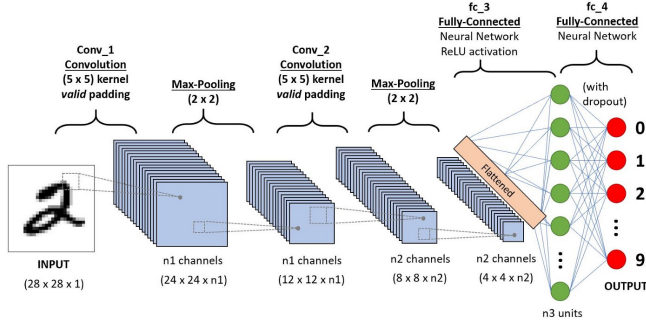


Fig. 5. CNN application example: classification for handwritten digits[7].

3. It can avoid overfitting problem, which is a common trouble in machine learning, because it takes the average of all the nodes that can get rid of the bias terms. Following is the typical structure of random forest algorithm[8].

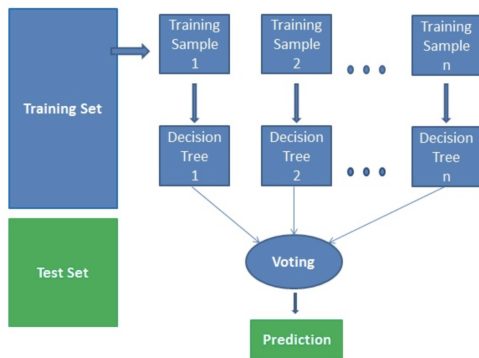


Fig. 6. Random forest structure[8].

4.3. Boosting

Boosting algorithm is a very popular algorithm in machine learning to convert weak learners to strong learners. A weak learner means this kind of learner is not sufficient or enough to train the model. If one uses the weak learning to train our model, the model will get very bad result. But boosting is a good algorithm that can convert this kind of weak learners into strong learners. It can solve this by using the average of good sub-results, weight all the sub-result, or take the most voted result. There are many types of boosting algorithms: AdaBoost (Adaptive Boosting), Gradient Boosting and XG-Boost. Here we used Gradient Boosting as our model. Typically, a gradient boosting involves three parts: a loss function that will be optimized, a weak learner that will make predictions and an additive model to add weak learners to minimize the loss function. Here is an example of how gradient boosting model works[9]:

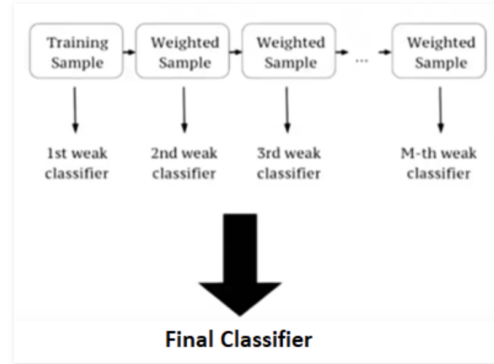


Fig. 7. How Gradient Boosting works[9].

4.4. Autoencoder

Auto encoder is a powerful method in neural network that uses a way to de-noise the data by dimensionally reduction the data. Along with the reduction side, a reconstructing side is learnt, where the autoencoder tries to generate from the reduced encoding a representation as close as possible to its original input[10].

5. EXPERIMENTS

In this section, we will introduce how we design, optimize and train our models respectively and introduce the results of our project.

5.1. CNN

For CNN, firstly, we trained with typical CNN model, which had two convolutional layers, corresponding maxpooling layer and four dense layers to generate the output. Because we converted the label into one-hot type, we chose categorical crossentropy as loss for backpropagation step. In order to reduce overfitting, we trained the model for 100 epochs and plotted the accuracy and loss verse epochs and we found that 15 epochs are enough for this training, as figure 8 shows.

Figure 8 also told us that this data is easy to get accurate result and maybe we can use more simplified model to reduce computational resource consuming. Then, we created a simple model with just 4 dense layers and trained it for 40 epochs and we get results as nice as before, and we will show that in the RESULT section.

5.2. Random Forest

For random forest, we used 26 features we extracted and set the random state as 23 (This random state shows a good result). Here we kept the rate of training set and testing set as 4:1, and randomly split these two parts from the given dataset. Notice that, the reason we set the parameter "max features" to 26 is that these 26 features are all very useful and powerful in

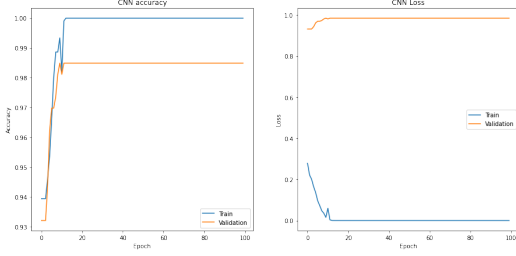


Fig. 8. Accuracy and loss verse epoch: after 15 epochs, nothing changes.

spectrum analysis so that we chose not to discard any one of them. Then we trained our data using the split training set and verified our result with the testing set.

5.3. Boosting

For the Boosting algorithm, we took the random rate as 23, number of estimators as 500, and learning rate as 0.1. Notice that here we set learning rate as a small number because our dataset is not very large. So if we use a large learning rate, this model would train too fast to lost some information and would show a bad result. Hence we used a lower learning rate to make our model learn slower that can fully use the information from our dataset.

5.4. Autoencoder

For this project, we designed a 4 layers linear autoencoder with loss function as categorical crossentropy. We chose to use a relatively trivial model for AE is because our data size was small (around 2000 data points) and we were just doing a linear classification. Thus, to avoid over-fitting and other issues that brought by the complex model implemented on a small dataset, linear AE was implemented in our design. It turned out that the output was pretty good. We tested 3 random states for 247, 245, 247 negative samples and 18, 20, 18 positive samples, respectively and the correctness for autoencoder are 98.1%, 98.5% and 98.9%.

5.5. Result

We did the cross validation for all models by splitting the data set into a training set and a test set in a 4:1 ratio. And we trained these models for several times with different random states for random initialization. Then we got the results and showed part of them in table 1. The results were expressed in 2*2 confusion matrix because we had two output states, positive or negative. At the same time we chose the best result and the worst result to show, where pos_neg means positive was diagnosed as positive and the others are like this.

In real life, the thing we care the most is the virus carrier who has not been diagnosed or detected. Also we don't want to treat people who don't carry the virus as patients, which

may cause trouble for them. So the focus will on pos_neg and neg_pos, in other words, misdiagnosis columns. All models with random state got nice results. Most of positives were detected, even for the worst case, only three positives were missed. For the best case, we correctly detected all of them. As we discussed before, CNN and SNN(simplified NN) gave us similar results and the result of SNN was even better.

We also noticed that all models gave us similar results, which might because of the assumption: can cough audio represents the patient? Is the patient's cough really different from that of a healthy person? The answer might be no; or because of feature extraction we did: can the features extracted from the audio represent the cough? These explanations are subject to further verification in the future.

Based on these results, we might applied these on mobile devices which can collect coughing sound and detect positives in public places like schools by analyzing their cough, and this will help to control the spread of this virus.

Table 1. Results of different models with random split

random state	model	pos_neg	pos_pos	neg_pos	neg_neg
1	CNN	3	15	2	245
	SNN	3	15	0	247
	RF	3	15	1	246
	Boosting	3	15	1	246
	AE	3	15	2	245
4	CNN	0	20	1	244
	SNN	0	20	1	244
	RF	0	20	1	244
	Boosting	0	20	1	244
	AE	0	20	4	241
5	CNN	1	17	1	246
	SNN	1	17	0	247
	RF	1	17	0	247
	Boosting	1	17	0	247
	AE	1	17	2	245

6. CONCLUSION

For the sake of finding some more efficient and cheaper COVID-19 detection methods to control spread of the virus, we implemented CNN, Random Forest, Boosting and Autoencoder to classify cough audios into positive or negative. These results that comes from our models gave us a good belief that there was a strong dependent relationship between cough and COVID-19. Also, those models can be applied in some real-life scenarios.

7. REFERENCES

- [1] Johns Hopkins University. Covid-19 dashboard by the center for systems science and engineering (csse) at johns hopkins university (jhu). In *Online Resource*, <https://gisanddata.maps.arcgis.com/apps/opsdashboard/index.html>, last visited June 12, 2020.
- [2] GlobeNewswire News Room. Curetis group company are genetics and bgi group collaborate to offer next-generation sequencing and pcr-based coronavirus (2019-ncov) testing in europe. In *Online Resource*, <https://www.globenewswire.com/news-release/2020/01/30/1977226/0/en/Curetis-Group-Company-Ares-Genetics-and-BGI-Group-Collaborate-to-Offer-Next-Generation-Sequencing-and-PCR-based-Coronavirus-2019-nCoV-Testing-in-Europe.html>, last visited June 12, 2020.
- [3] S. Doshi. Music feature extraction in python. In *Online Resource*, <https://towardsdatascience.com/extract-features-of-music-75a3f9bc265d>, last visited May 12, 2020.
- [4] S. Doshi. Classification of music into different genres using keras. In *Online Resource*, <https://medium.com/@sdoshi579/classification-of-music-into-different-genres-using-keras-82ab5339efe0>, last visited May 12, 2020.
- [5] Haneya N. Qureshi Usama Masood Sajid Riaz Kamran Ali Charles N. John Muhammad Nabeel Iftikhar Hussain Ali Imran, Iryna Posokhova. Ai4covid-19: Ai enabled preliminary diagnosis for covid-19 from cough samples via an app. *arxiv*, 2020.
- [6] Y. Chen, T. Krishna, J. S. Emer, and V. Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE Journal of Solid-State Circuits*, 52(1):127–138, 2017.
- [7] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way. In *Online Resource*, <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>, last visited May 12, 2020.
- [8] Avinash Navlani. Understanding random forests classifiers in python. In *Understanding Random Forests Classifiers in Python*, <https://www.datacamp.com/community/tutorials/random-forests-classifier-python>.
- [9] Prateek Aggarwal. Ml — xgboost (extreme gradient boosting). 2020. <https://www.geeksforgeeks.org/ml-xgboost-extreme-gradient-boosting/>.
- [10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.

Individual contributions

- Jiajun Du
Perform the models of Random Forest and Boosting. Illustration of the result for these two models. Show the graphs of original audio data and short-time fourier transform spectrum.
- Kunpeng Liu
Data preprocessing: process audio files and extract features such as rmse, mfcc, which can represent the characteristic of cough;
Design CNN model and optimize the model to train the data set.
- Haoming Zhang
Design, optimize and train autoencoder model from given dataset
literature search: AI4COVID19
Background search for COVID-19

Replies on critical reviews

Critical review from team 3:

- More introduction to related works

After searching online, we found a paper “ai4covid19” did similar to our project and added to our literature review.

- More explanation about feature extraction

In section 3. dataset and feature part, we explained how we extracted data and how we used them in detail.

- Compare the model performance with some visualization ways

We made a table to illustrate the performance for all the models in a different random state and it turns out all models worked pretty good at classifying!

- Show more details about the optimization and the model

Since the time limitation of the presentation, we did not explain how we optimize our model and how we design our model. In the models and design part of this paper, we illustrate in detail how we choose, optimize, and design our model.

Critical review from team 62:

- The project explores many different models on the task of detecting the coughing of COVID-19, including CNN, simplified NN, Random Forest, Boosting and AutoEncoder. In the presentation, the authors also gave an intuitive comparison among all the models implemented.

- According to the comparison, all the models show great performance on the task with a high accuracy. The result also shows that different models obtained similar performance indicating that data extraction could result in a bigger performance improvement rather than optimizing the model.

- However, in order to further compare the performance, it would be better that the presentation could compare more aspects than the accuracy, for example, the model size, training time, model complexity, etc.

- Also, to fully understand the models adopted by the project, the authors should also give more specific details. For example, the author suggested that they introduced several kinds of features, but how the features combined and reshaped to feed into the models is not shown in the slide.

Besides, more details about the hyperparameters about the models in the slides would also be nice.

Our response:

Actually we didn't include much details in the presentation due to the limitation of time, but we have shown details in our paper. I'm sure you can get what you want to see.

Also, we didn't compare much aspects besides accuracy, because the models we used are simple. My logic is to get similar results through different models to get our conclusion, hence, we don't want to add additional comparison. We will do it if necessary. For example, in CNN model, we found we can train the model for 15 epochs to get nice results, so we simplified the model.

Critical review from team 71:

- More literature resources:

In the reference part, we list all the specific references that we used in this report.

- Lack of Covid-19 cough data:

The data is given by TAs on Canvas, which itself is a limited amount.

- More graphs to show the result:

In the conclusion part, we add a graph and a table to show our results.

- Why choose this confusion matrix:

Because we really care about the negative and positive accuracy separately.

- More literature resources:

In the reference part, we list all the specific references that we used in this report.

- Any hyperparameters:

Yes, we use some hyperparameters in deep learning models.

- More graphs on overfitting:

We add the overfitting plot in the report.