Group 3 Reply to Review

- How do you handle the imbalance between the negative and positive training data? Because negative-COVID coughs are overly represented in the dataset, the model can just “guess” negative class and still obtain high accuracy.
  A: We tried to use part of the negative training dataset as input initially, and increase the size of the negative training dataset gradually, then plot and observe the change of accuracy. Indeed, the performance is degraded. To improve, we weight positive label’s contribution to the cost function inversely proportional to the frequency of the label. However, this method causes the significant overfitting to the positive signals. The best method is still seeking more available positive training datasets.

- In the No.10 slides, the speaker gave a comprehensive speech on the data preprocessing operation. Since we were using the same dataset and we found some of the recordings contained little information or could not be read, so we are wondering how the noise data was being sifted.
  A: The unsatisfied recordings are detected by observing the size of file in the folders, and they are directly removed from the dataset manually.

- What and why are the choices of loss functions for the models? Please also plot the loss change during the training to demonstrate the training process in the final report.
  A: In our model, the loss function is binary_crossentropy, which calculates a score that summarizes the average difference between the actual and predicted probability distributions for predicting class 1. It’s the best option for a 2-class classification problem.

- They showed the prediction accuracy on the entire dataset. How about prediction accuracy on positive or negative data in the testing set? Or say the true negative and true positive rate on a testing dataset?
  A: We think it is rather important to show the prediction accuracy for the entire dataset. For true negative and positive data, we have done concrete analysis on it as well. The results turned out to align well with the whole dataset.

- All three models seem to have good performance at the first epoch (>90% accuracy). Does this happen by chance? You could consider try difference initialization and test the models using loss, true positive rate, and true negative rate as criterions.

A: We have done several runs of testing. We only showed the best result we have so far for presentation purposes. It does not happen by chance.

- Literature survey is a bit poor, no real connections/ties to any previous works. Without specific examples of previous works in machine learning (e.g. machine learning for general respiratory disease or audio classification?), it seems to be more of repetition/expansion of the background information.
  A: We did not find too much information about these cases with ML methods. So we expand it and show some traditional methods about processing the respiratory disease.

COVID-19 RISK FORECASTING BASED ON COUGH SIGNAL

Boxuan Xiao, Rui Yang, Xinqiao Zhang

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

ABSTRACT

In 2020, with covid-19 infectious disease pandemic worldwide. How to detect and judge suspected cases as soon as possible is a problem that needs to be solved urgently. We conducted experiments on different neural networks, analyzed their experimental results on this data set, and made corresponding improvements.

Index Terms— Deep Neural Network, STFT, Mel Filterbank, LSTM

1. INTRODUCTION

In 2020, the covid-19 virus is raging all over the world, and scholars and doctors from various countries hope to diagnose and treat the disease faster.[1] Improving the speed and accuracy of detecting new coronavirus is currently urgently needed. The most commonly used method is the new crown virus-specific reverse transcription polymerase chain reaction (RT-PCR). Testing can take up to two days to complete, and repeated testing may be required to rule out potential false negative results. The medical community believes that chest CT scan is a useful tool for diagnosing suspected cases of new coronavirus. However, in the specific case of patients suffering from other types of lung disease, a single CT scan cannot exclude the new coronavirus[2]. In our project, we use AI algorithm to quickly diagnose new coronavirus positive patients. We use audio data sets that use positive and negative coughs to train and test the AI model. The new AI system helps to quickly diagnose patients with new coronavirus. Machine learning and deep learning technologies are very powerful tools in the data age. Because it greatly reduces the consumption of human resources, we can solve the problems we encountered more quickly and digitally. Therefore, in the face of the current severe global epidemic, it is very effective to use this technology to respond to such large-scale testing. We can detect suspected cases more quickly and gain valuable time for subsequent treatment and epidemic prevention. At the same time, using machine learning technology, we can better reduce direct contact between people to reduce the risk of infection.Our method can be used as a convenient and large-scale preliminary judgment, but it does not have medical reliability, but only provides a good preliminary prediction for patients and doctors to carry out the next diagnosis and treatment to shorten the early diagnosis time.

2. RELATED WORK

2.1. Background

The virus mainly spreads through the respiratory tract, so how to detect suspected cases through respiratory symptoms is extremely important. Our project uses machine learning to detect suspected cases by detecting coughing sounds for further protection

2.2. Traditional methods

Traditionally, doctors use the learned professional knowledge and face-to-face experience to initially determine whether the patient has related illnesses. However, this method has low efficiency and its accuracy cannot be quantified. At the same time, because of the need for face-to-face consultation, it greatly increases the risk of transmission between doctors and patients

2.3. CT and DNA methods

With the development of medical technology, humans are gradually using scientific techniques, such as CT imaging[3], nucleic acid detection and other medical imaging technologies and DNA technology to make more accurate diagnosis. However, although these diagnostic methods can be used for more effective diagnosis, due to the long processing time, large-scale outbreaks cannot be diagnosed quickly

3. DATASET AND FEATURE EXTRACTION

3.1. Information of Cough Sound Signal Data

The mainly involved dataset in this project is provided by Noise Lab in University of California San Diego. The dataset includes 1287 signals of coughs from negative cases, and 89 signals of coughs from positive cases. We also self-record three cough signals from our group members as real-life test cases. The signals are in .wav format. This format is uncompressed and can be loaded into RAM rapidly, then no further format convert is needed. The sample rate of signals is 48kHz, and the bit rate is 768kbps. We made a tradeoff between efficiency and accuracy by reducing the size of the data. The signals are resampled to 16kHz, then the input shapes are decreased from 48000 to 16000. 34 signals are unsatisfiable and contains insufficient information, they are detected by the size of file and removed manually. 80% of dataset is assigned to training size, 20% dataset are assigned to validation set.

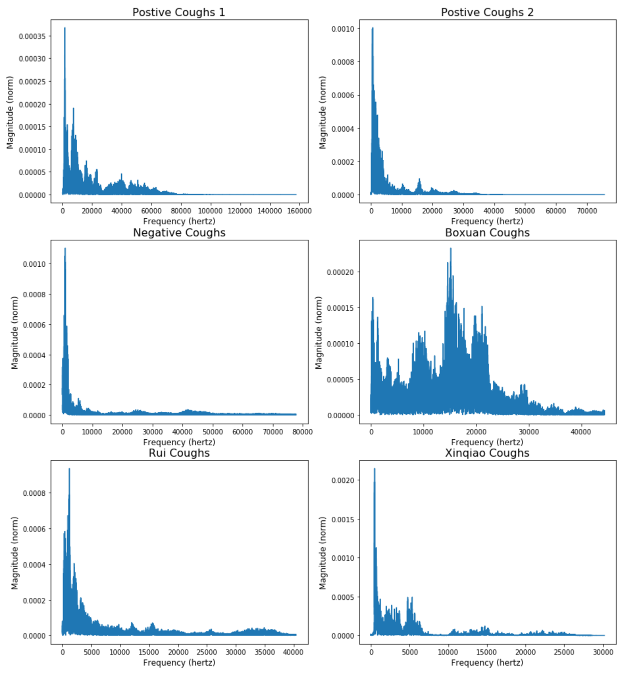


Fig. 1. Results of STFT in frequency domain. Include 2 positive example cases, 1 negative example case and self-record cough from group members.

3.2. Threshold Envelope and Single Cough Detection

Most given signals are formed by multiple coughs and the dead zones containing no information. To extract the principle parts, we implement a rolling window to build a signal envelop[4], which detects single cough and removes the dead zones. Any points lower than the threshold (set to 100 in our experiment) are dropped. However, the waveform always crosses the x-axis. To avoid wrongly dropping out small values inside a single cough range, we calculate the mean of absolute value of points in the rolling window. For the points above the threshold, we keep and append them into a new list, and this new list will replace the original waveform. Then no dead zones exist in processed signals.

3.3. Feature Extraction and Mel Scale

The frequency domain analysis, which shows the energy distribution of the signals over frequencies, contains several important characteristics of signal that can be utilize in deep learning neural networks. The Short Term Fourier Transform (STFT) converts time series waveforms to visible frequency domain plots as the Figure shows. Generally, human being is unsensitive to the high pitch sounds, and it's impossible for an adult to emit a sound in 2000Hz. Thus, for a cough classification problem, low frequency parts are supposed to have heavier weight than high frequency parts, and the STFT plot proves that. In the plot we find the majority part of signal are in low frequency range.

Instead of describing the signal in normal logarithmic scale, we describe the frequency characteristic of signal in Mel scale, which is not linear but has more importance on lower frequency. The following equation can convert frequency from Hertz to Mel scale[5].

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \tag{1}$$

In high frequency range, we find it requires increasingly large intervals in Hz to produce one unit increment of Mel scale. It satisfies our demand of putting more importance on low frequency characteristics. The transition is applied by implementing numbers of triangular filters (128 filters in this experiment) to the power spectrum to extract frequency bands. To avoid storing excessive processed signal offline, we call the function in Kapre Audio Preprocessors library developed by Keunwoo Choi. It enables the Mel conversion to be implemented real-timely as a Keras layer[6].

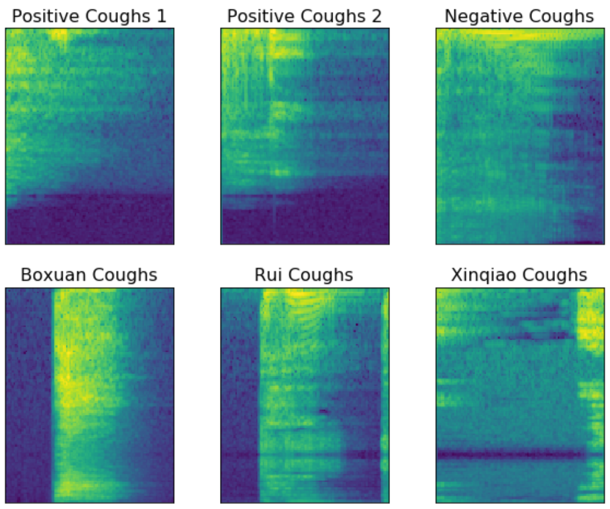
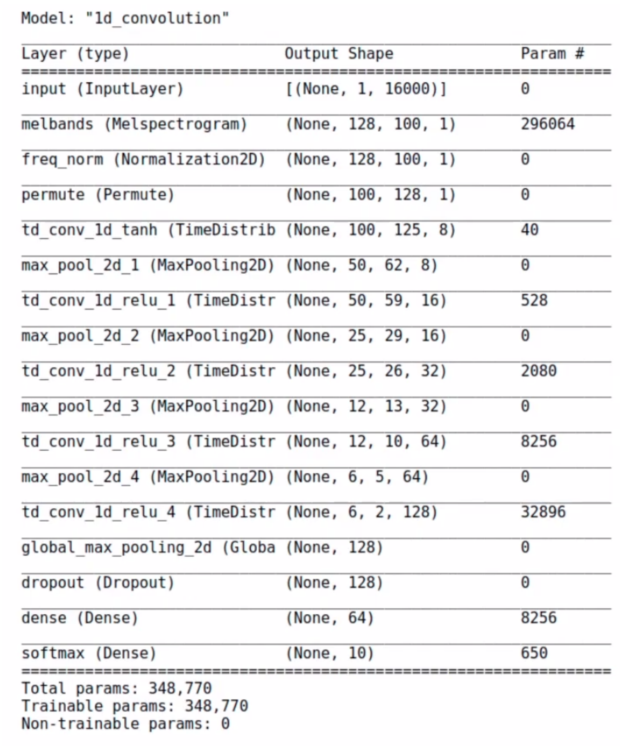


Fig. 2. Spectrograms of processed examples of signal in Mel Scale. Windows length is set to 400 samples, and step size is set to 160 samples. 100 time-series samples are shown within each spectrogram.

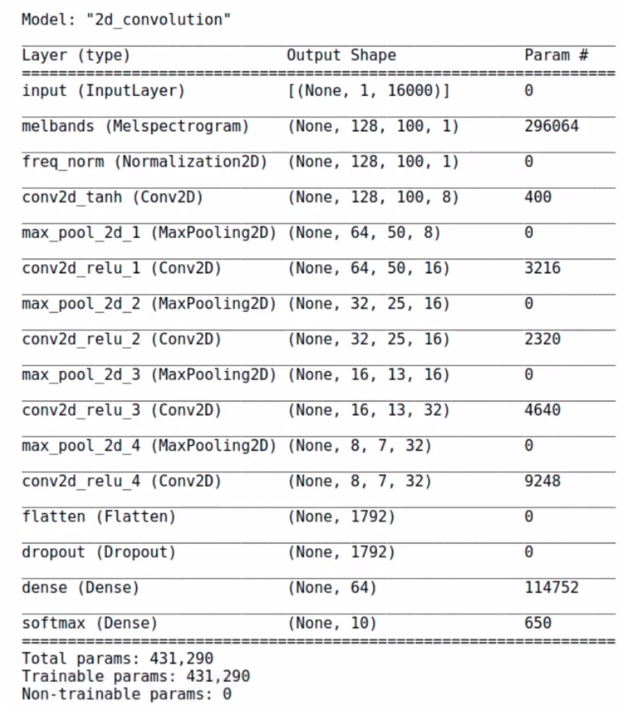
4. MODEL AND EXPERIMENT

Figure 3.3,Figure 4 and Figure 5 show the configuration of our 1D model, 2D model and LSTM model respectively. Note that for all three models, we use kapre package for Mel-spectrogram and Normalization2D layers. And for 1D and



```
Model: "1d_convolution"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input (InputLayer)           [(None, 1, 16000)]        0
_________________________________________________________________
melbands (Melspectrogram)    (None, 128, 100, 1)       296064
_________________________________________________________________
freq_norm (Normalization2D)  (None, 128, 100, 1)       0
_________________________________________________________________
permute (Permute)            (None, 100, 128, 1)       0
_________________________________________________________________
td_conv_1d_tanh (TimeDistrib (None, 100, 125, 8)       40
_________________________________________________________________
max_pool_2d_1 (MaxPooling2D) (None, 50, 62, 8)         0
_________________________________________________________________
td_conv_1d_relu_1 (TimeDistr (None, 50, 59, 16)        528
_________________________________________________________________
max_pool_2d_2 (MaxPooling2D) (None, 25, 29, 16)        0
_________________________________________________________________
td_conv_1d_relu_2 (TimeDistr (None, 25, 26, 32)        2080
_________________________________________________________________
max_pool_2d_3 (MaxPooling2D) (None, 12, 13, 32)        0
_________________________________________________________________
td_conv_1d_relu_3 (TimeDistr (None, 12, 10, 64)        8256
_________________________________________________________________
max_pool_2d_4 (MaxPooling2D) (None, 6, 5, 64)          0
_________________________________________________________________
td_conv_1d_relu_4 (TimeDistr (None, 6, 2, 128)         32896
_________________________________________________________________
global_max_pooling_2d (Globa (None, 128)               0
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 64)                8256
_________________________________________________________________
softmax (Dense)              (None, 10)                650
=================================================================
Total params: 348,770
Trainable params: 348,770
Non-trainable params: 0
```

Fig. 3. 1D model

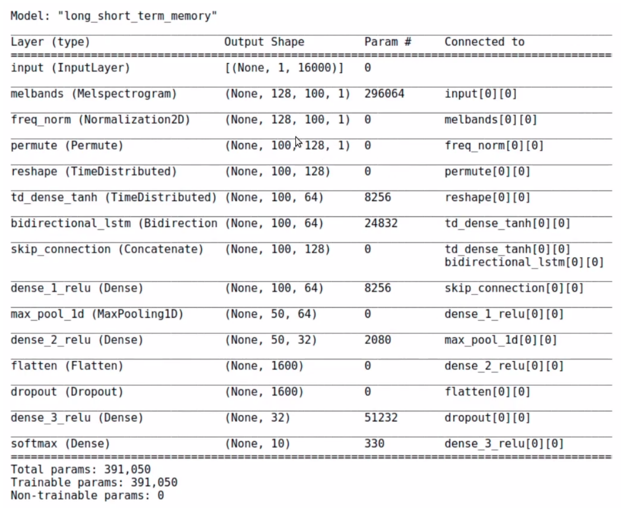


```
Model: "2d_convolution"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input (InputLayer)           [(None, 1, 16000)]        0
_________________________________________________________________
melbands (Melspectrogram)    (None, 128, 100, 1)       296064
_________________________________________________________________
freq_norm (Normalization2D)  (None, 128, 100, 1)       0
_________________________________________________________________
conv2d_tanh (Conv2D)         (None, 128, 100, 8)       400
_________________________________________________________________
max_pool_2d_1 (MaxPooling2D) (None, 64, 50, 8)         0
_________________________________________________________________
conv2d_relu_1 (Conv2D)       (None, 64, 50, 16)        3216
_________________________________________________________________
max_pool_2d_2 (MaxPooling2D) (None, 32, 25, 16)        0
_________________________________________________________________
conv2d_relu_2 (Conv2D)       (None, 32, 25, 16)        2320
_________________________________________________________________
max_pool_2d_3 (MaxPooling2D) (None, 16, 13, 16)        0
_________________________________________________________________
conv2d_relu_3 (Conv2D)       (None, 16, 13, 32)        4640
_________________________________________________________________
max_pool_2d_4 (MaxPooling2D) (None, 8, 7, 32)          0
_________________________________________________________________
conv2d_relu_4 (Conv2D)       (None, 8, 7, 32)          9248
_________________________________________________________________
flatten (Flatten)            (None, 1792)              0
_________________________________________________________________
dropout (Dropout)            (None, 1792)              0
_________________________________________________________________
dense (Dense)                (None, 64)                114752
_________________________________________________________________
softmax (Dense)              (None, 10)                650
=================================================================
Total params: 431,290
Trainable params: 431,290
Non-trainable params: 0
```

Fig. 4. 2D model

2D model, we use some normal layers like dense, maxpool, conv1d and conv2d layers.

Figure 6 shows our experiment among different percent-



```
Model: "long_short_term_memory"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input (InputLayer)              [(None, 1, 16000)]   0
__________________________________________________________________________________________________
melbands (Melspectrogram)       (None, 128, 100, 1)  296064      input[0][0]
__________________________________________________________________________________________________
freq_norm (Normalization2D)     (None, 128, 100, 1)  0           melbands[0][0]
__________________________________________________________________________________________________
permute (Permute)               (None, 100, 128, 1)  0           freq_norm[0][0]
__________________________________________________________________________________________________
reshape (TimeDistributed)       (None, 100, 128)     0           permute[0][0]
__________________________________________________________________________________________________
td_dense_tanh (TimeDistributed) (None, 100, 64)      8256        reshape[0][0]
__________________________________________________________________________________________________
bidirectional_lstm (Bidirection (None, 100, 64)      24832       td_dense_tanh[0][0]
__________________________________________________________________________________________________
skip_connection (Concatenate)   (None, 100, 128)     0           td_dense_tanh[0][0]
                                                                 bidirectional_lstm[0][0]
__________________________________________________________________________________________________
dense_1_relu (Dense)            (None, 100, 64)      8256        skip_connection[0][0]
__________________________________________________________________________________________________
max_pool_1d (MaxPooling1D)      (None, 50, 64)       0           dense_1_relu[0][0]
__________________________________________________________________________________________________
dense_2_relu (Dense)            (None, 50, 32)       2080        max_pool_1d[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 1600)         0           dense_2_relu[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 1600)         0           flatten[0][0]
__________________________________________________________________________________________________
dense_3_relu (Dense)            (None, 32)           51232       dropout[0][0]
__________________________________________________________________________________________________
softmax (Dense)                 (None, 10)           330         dense_3_relu[0][0]
==================================================================================================
Total params: 391,050
Trainable params: 391,050
Non-trainable params: 0
```

Fig. 5. LSTM model

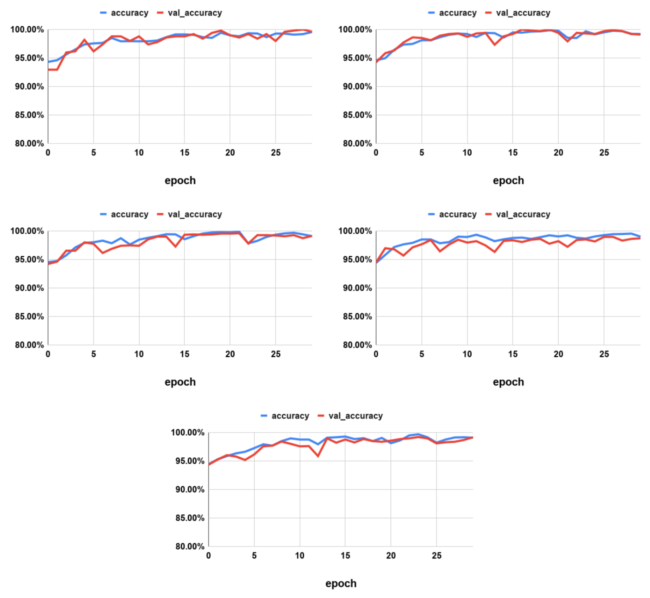


Fig. 6. Different percentage of training data and validating data. Traing data/ validating data = (a) 0.9/0.1 (b) 0.8/0.2 (c) 0.7/0.3 (d) 0.6/0.4 (e) 0.5/0.5.

age of training data and validating data. The red line is test accuracy . From the figure we can see that our LSTM model can get a very good result for all experiments. Especially for (b), it get best performance and it reaches the highest accuracy in just a few epochs.

We also apply scalability analysis for our LSTM model. We run it for 100 epochs and it turns out to be good during the 100 interactions, and it will not be overfitting, which means this model has a very good scalability and it can apply to different data. Figure 7 shows our result for 100 epochs.

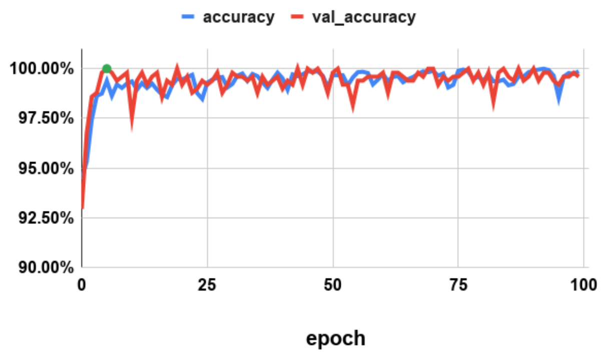


Fig. 7. scalability analysis for LSTM model

5. CONCLUSION

Our project is a good realization of the initial judgment of covid-19 by coughing audio data, which greatly speeds up the diagnosis process and reduces labor costs and reduces the risk of virus transmission. It has a good reference for subsequent related research and experiments

6. INDIVIDUAL CONTRIBUTION

Boxuan Xiao : Doing research on different machine learning models, find the most adaptive and fitting one. Apply parameters combination tests, improve the performance of the model. Implement digital signal process techniques to preprocess the dataset. Find cutting edge papers related to our topic, learn and use the paper proposed methods for reference.

Rui Yang :Help collecting the data. Do the pretreatment about the data. Doing research and applying some methods about feature extraction. Learning about the model and try to apply or improve these models on our data.

Xinqiao Zhang : Do all the experiment part for 1D, 2D and LSTM model. Get the results and plot the data. Do scalability analysis and try to improve the LSTM model accuracy. Write some part of the report. Do some research and initially trying to test if PCA or other algorithms works for COVID-19 data. Add one potential feature extraction method for the model. Briefly answered some questions about the project progress report.

7. REFERENCES

[1] WHO. *Coronavirus disease 2019 (COVID-19) Situation Report – 41; 2020.*

[2] Nishiura H, Kobayashi T, Yang Y, Hayashi K, Miyama T, Kinoshita R, et al. *The rate of underascertainment of novel coronavirus (2019-nCoV) infection: Estimation using Japanese passengers data on evacuation flights*. 2020.

[3] Adam Bernheim, Xueyan Mei, Mingqian Huang, Yang Yang, Zahi A. Fayad, Ning Zhang, Kaiyue Diao, Bin Lin, Xiqi Zhu, Kunwei Li, Shaolin Li, Hong Shan, Adam Jacobi, and Michael Chung. Chest ct findings in coronavirus disease-19 (covid-19): Relationship to duration of infection. *Radiology*, 295(3):200463, 2020. PMID: 32077789.

[4] R. Mignot and V. Välimäki. True discrete cepstrum: An accurate and smooth spectral envelope estimation for music processing. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7465–7469, 2014.

[5] O. Farooq and S. Datta. Mel filter-like admissible wavelet packet structure for speech recognition. *IEEE Signal Processing Letters*, 8(7):196–198, 2001.

[6] Keunwoo Choi, Deokjin Joo, and Juho Kim. Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras. *CoRR*, abs/1706.05781, 2017.