INTERSPEECH 2021
30 August – 3 September, 2021, Brno, Czechia

Transfer Learning and Data Augmentation Techniques to the COVID-19 Identification Tasks in ComParE 2021

Edresson Casanova[1], *Arnaldo Candido Jr.*[2], *Ricardo Corso Fernandes Jr.*[2], *Marcelo Finger*[3], *Lucas Rafael Stefanel Gris*[2], *Moacir A. Ponti*[1], *Daniel Peixoto Pinto da Silva*[2]

[1] Instituto de Ciências Matemáticas e de Computação, University of São Paulo, Brazil
[2] Federal University of Technology – Paraná, Brazil
[3] Dept. of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, Brazil

edresson@usp.br

Abstract

In this work, we propose several techniques to address data scarceness in ComParE 2021 COVID-19 identification tasks for the application of deep models such as Convolutional Neural Networks. Data is initially preprocessed into spectrogram or MFCC-gram formats. After preprocessing, we combine three different data augmentation techniques to be applied in model training. Then we employ transfer learning techniques from pretrained audio neural networks. Those techniques are applied to several distinct neural architectures. For COVID-19 identification in speech segments, we obtained competitive results. On the other hand, in the identification task based on cough data, we succeeded in producing a noticeable improvement on existing baselines, reaching 75.9% unweighted average recall (UAR).

Index Terms: computational paralinguistics, COVID-19, deep learning, data augmentation, transfer learning

1. Introduction

Automated analysis of audio signals turned out to be a promising path for the screening of respiratory diseases [1, 2]. One year after COVID-19 was officially declared a global pandemic, there is a huge interest on improving such tools, allowing alternative forms of identification of suspicious cases of infection.

The Interspeech Computational Paralinguistics ChallengE (ComParE) 2021 proposes two competitions related to detecting COVID-19 from audio samples. Such samples represent speech and cough from both healthy and infected speakers. The COVID-19 Speech Sub-Challenge (CSS) offers 3.24 hours of audio recordings containing speech samples, while the COVID-19 Cough Sub-Challenge (CCS) provides 1.63 hours of cough samples.

Our contribution to ComParE 2021 explores four architectures based on convolutional neural networks aiming at detecting COVID-19 in both CSS and CCS. In particular, we investigate employing transfer learning, a well-established technique to address data scarceness and adaptation between different datasets [3], which has been used successfully in previous editions of ComParE [4, 5, 6, 7]. Thus our proposal studies transfer learning from Pretrained Audio Neural Networks (PANNs) [8], which are models trained on millions of audio samples. In addition, we explore three different data augmentation techniques. The main contributions can be summarized as: (i) investigating large-scale pretrained audio neural networks for the identification of COVID-19; (ii) improving the generalization of deep models via three data augmentation techniques, i.e. Mixup [9], SpecAugment [10] and additive noise data augmentation [11].

2. Experimental Framework

First, we present the original datasets (section 2.1), noting that there are insufficient data for a deep learning approach. Due to this scarcity, we applied three techniques for data augmentation: Noise Data Augmentation (section 2.2), SpecAugment (section 2.3) and Mixup (section 2.4), in that order and combining the three methods. Finally, we also explore transfer learning to address data scarcity (section 2.5).

2.1. Datasets

The datasets for the sub-challenges were extracted from the COVID-19 Sound database [12, 13, 14]. Two groups were defined, audio recordings from subjects with COVID-19 (patient group) and from individuals not infected and therefore without COVID-19 symptoms (control group).

Regarding the CSS, 893 audio clips were gathered from 366 speakers and different languages. There are 3.24 hours of recordings, with duration varying between 3.6 and 30.1 seconds. A total of 315 instances for training: 243 from the control group and 72 for patient group. A development set is also available with 295 samples: 153 for control and 142 for patients. The test set consists of 283 audios without public labels [14].

For the CCS, 929 cough audio instances were obtained from 397 speakers, resulting in 1.63 hours. The training set presents 215 and 71 audios for control and patients, respectively (286 in total). The development set contains 231 audios (183 for control and 48 for patients). The test set is composed of 208 unlabeled audio clips [14].

Both the CSS and CCS audios are sampled at 16khz. It is known the dataset contains multiple instances per speaker. In addition, the training, development, and test sets do not share speakers, thus they are speaker independent [14].

2.2. Noise data augmentation

We use the additive noise method which is popular in speech processing [11]. For this purpose, we use the MUSAN corpus [15], which is composed of three subsets: the noise subset, which contains approximately 6 hours of noise, such as ambient noise and dialtones; the music subset, which contains approximately 42 hours of music and; the speech subset, containing 60 hours of human speech. We choose to use only the noise subset, because it contains ambient noise that represents most of the noise found in the datasets used in this paper. A noise sample in every training step is chosen randomly and is added to the original signal with a signal to noise ratio random varying from 0 to 15dB, similar to the proposal of [16].

http://dx.doi.org/10.21437/Interspeech.2021-1798

2.3. SpecAugment

SpecAugment [10] is an augmentation method originally proposed for automatic speech recognition. It consists of masking a log-mel spectrogram in the frequency and time domain [8]. The frequency masking is applied so that f consecutive mel channels $[f0, f0 + f]$ are masked, where f is chosen from a uniform distribution ranging from zero to a previously defined value that indicates the maximum masking width. Finally, $f0$ is chosen at random from $[0, v - f]$, where v is the number of mel channels [10]. Masking in time domain is similar to frequency masking, but along the time axis [8]. Similar to [8], we used two masks in the frequency domain, with a maximum width of eight mel channels and two masks in the time domain with a maximum width of 64 frames.

2.4. Mixup

Mixup is a technique proposed by [9] to address overfitting and sensitivity, using adversarial examples in deep neural networks. The authors showed that this method improves the generalization of deep models. In this method, at each training step, the neural network is trained on a convex combination of two inputs and their corresponding labels [6], creating interpolations between pairs of instances. Given two inputs x_i and x_j and their corresponding classes y_i and y_j, their resulting Mixup $(\widetilde{x}, \widetilde{y})$ is defined in Eq. (1).

$$\begin{aligned} \widetilde{x} &= \lambda x_i + (1 - \lambda) x_j \\ \widetilde{y} &= \lambda y_i + (1 - \lambda) y_j, \end{aligned} \tag{1}$$

The parameter λ in (1) is sampled for each pair (i, j) from a Beta Distribution with parameters α, $\lambda \sim Beta(\alpha, \alpha)$, where $\alpha \in (0, \infty)$ [9]. We tested several possible Beta distributions with $\alpha \in (0, 1)$, following guidelines of [6].

2.5. PANNs: AudioSet Pre-training

Transfer learning has shown to be a promising technique in several tasks [17, 18, 19], such as computer vision [18], where pretrained models are used as feature extractors in images [20], videos [21], and show to be useful when different datasets are used as source and target tasks [3]. Finally, in the audio domain, transfer learning obtained from images is also used in audio pattern recognition tasks [4]. The Pretrained Audio Neural Networks (PANNs) [8] were proposed in order to support several tasks such as audio tagging, acoustic scene classification, music classification, speech emotion classification and sound event detection.

PANNs used in this were trained in AudioSet [22], a dataset containing approximately 1.9 million audio samples from 527 different classes. The authors explored different convolutional topologies, such as VGG-like CNNs [23], MobileNets [24] and ResNets [25]. As a result, the authors reported the best models, which were trained using as input log-mel spectrograms extracted from audios sampled at 32 kHz. The log-mel spectrogram was extracted using Fast Fourier Transform (FFT) [26], with Hamming window 1024, hop length 320 and 1024 FFT components. In addition, 64 mel filter banks were applied to calculate the log-mel spectrogram. Overall, there is good empirical evidence in favor of applying PANNs in audio pattern recognition tasks [8], especially under limited availability of training data. In this work we explore transfer learning from three PANNs, namely CNN14, ResNet-38 and MobileNetv1.

3. Experiments

In order to build a classification model for COVID-19 detection, four scenarios of data augmentation and neural network topologies were investigated. Firstly, we present the four experiments in section 3.1. Then, detailed explanation regarding implementation is given in section 3.2. Finally, we describe the choice of hyper-parameters used to build each model in section 3.3.

3.1. Proposed Experiments

Each experiment is based on a distinct neural network topology and investigates the use of transfer learning and different data augmentation approaches. To facilitate reproducibility, the source code used in each experiment is available on GitHub[1]. The experiments were carried out as follows:

- **SpiraNet:** uses the SPIRA-project topology proposed specifically for detecting respiratory insufficiency in speech audio samples represented as Mel Frequency Cepstral Coefficients (MFCCs) [27]. The authors proposed a convolutional neural network to identify respiratory insufficiency in infected patients with COVID-19 in a scenario of high environmental noise, achieving promising results ($\sim 91\%$ accuracy). In this experiment, we use the same topology. Given the dataset difference, we searched for alternative strategies and fine-tuned hyper-parameters, namely the kernel size, convolutional dilatation, dropout, number of fully connected layer neurons after the last convolutional layer (FC dim), learning rate, weight decay and optimizer for each of the dataset (CSS, CCS). We use MFCCs with the same parameters reported in [27]. For this experiment we *do not* use noise data augmentation, Mixup and SpecAugment.
- **CNN14:** this experiment explores transfer learning from the PANN CNN14 model [8]. We adjusted the following original hyper-parameters in order to increase performance in the CSS and CCS datasets: Mixup α, learning rate, weight decay and optimizer for each of the datasets. The input is composed of log-mel spectrograms. We used the same parameters as the PANNs to extract the log-mel spectrogram (described in Section 2.5), except for the sampling rate, which was 16 kHz as in CSS and CCS datasets. This models makes use of noise data augmentation, Mixup and SpecAugment.
- **ResNet-38:** it is similar to CNN14, but exploring transfer learning of the PANN ResNet-38 [8].
- **MobileNetv1:** also similar to CNN14, but exploring transfer learning from PANN MobileNetv1 [8].

Each proposed experiment was executed using three different configurations: simple holdout, windowed holdout and K-fold cross validation. In all the cases, classifier ensembles were created.

Regarding the simple holdout method, the official training and development sets were used. This method was applied in two stages. First, the hyper-parameters of the four neural networks were fine-tuned, one at a time, using a fixed random seed. In the second phase, each experiment was carried out five times using five different random seeds and an ensemble was created, performing vote by sum of probabilities. We report only the results for the second phase.

[1] https://github.com/Edresson/SPIRA-ComParE2021

For the windowed holdout, the procedure is similar to the simple holdout, however we do not use the entire sample of audios for training the networks. In this approach, training is performed by selecting a random window from the original input audio, the window size varies according to the dataset used (detailed in Section 3.2). For development and test, the classifiers received all possible windows from the audio considering the hop size as window size (no overlapping) and their results are combined in the voting process.

Finally, in the k-fold cross validation, we merged the training and development sets, using five folds to evaluate each experiment. In this approach, each fold is used in the training of ensembles rather than single models. Each ensemble is composed of the five models (obtained with five different random seeds initialization), yielding 25 models over all folds. The development set results are obtained analysing each fold individually, by voting. In the test set, however, we perform a different approach, inspired in [4], joining all 25 models to vote. This approach has the main advantage of covering more samples from the training and development sets in order to decide the class. As a drawback, the speaker independence assumption presented in Section 2.1 may be compromised. Despite this, in [4] the authors showed that this approach was effective in a scenario without the independence of speakers between the train and development set, turning out to be the winner of the ComParE 2020 Mask Sub-Challenge.

3.2. Experiment Implementation Details

In all experiments, our models are trained for 100 epochs. We use the Binary Cross Entropy function [28] as loss for training experiments based on Mixup. On the other hand, for experiments that do not use Mixup, the loss function used was the average of the Binary Cross Entropy calculated individually for each class. Finally, in all experiments we choose the best checkpoint using the average of the Binary Cross Entropy calculated individually for each class in the development set.

The audio duration was set as the maximum duration on the dataset for simple holdout and cross validation experiments in order to standardize the audio duration, allowing network training and inference. Zero padding was used to adjust audios duration when needed, injecting silence in the samples. For windowed holdout, an audio window is chosen at random and the window size is three seconds for the CSS dataset and two seconds for the CCS dataset. These values were chosen because they represent the rounded down value of the smallest audios in each of the datasets.

Because the dataset has significant class imbalance, for the k-fold cross validation, each fold is generated using proportional stratified sampling. Batches, however, uses a different method, either for cross validation or other holdout approaches. Weighted random sampling [29] is used to build a batch containing a balanced number of instances for each class. In this technique, each instance receives a weight indicating its probability to be selected [30]. The inverse frequency of each class is used to defined the weights of its instances.

All of our models were implemented using Pytorch 1.6.0 [29] and trained on an NVIDIA Titan RTX GPU with 24GB of memory on a server with Intel (R) Core (TM) i9-10900 CPU and 128GB of RAM. In addition, all models were trained with a batch size of 16 and using Noam's decay scheme [31] applied to every 500 steps.

3.3. Models hyper-parameters

All the hyper-parameter values presented in this section were manually adjusted, one at a time for both simple and windowed holdout approaches. On the other hand, K-fold procedure uses the same hyper-parameter values as in holdout. For this adjusts, we used the development set loss.

Regarding **CSS**, the following hyper-parameters were chosen in both holdout and k-fold evaluation. Adam [32] was used for experiments 1, 3 and 4, while RAdamc [33] was used in experiment 2. The initial learning rate were defined as 0.1 for experiment 1 and 0.001 for the others. Weight decay of 0.01 was used in experiment 1 and 4, while 1e-05 was used for the others. Mixup $\alpha = 1$ were used in experiment 2, 3 and 4. Experiment 1 had other parameters adjusted, namely in FC dim 100, dropout rate 0.7, convolutional dilatation 2×1 and kernel size 5×1. The remaining experiments use transfer learning and these hyper-parameters are kept at the default values.

In the windowed holdout approach, we used Adam optimizer for experiment 3 and AdamW optimizer [34] for the others. The learning rate was set to 0.01 in experiment 1 and 0.001 for the others. Experiments 1 and 4 had weight decay of 0.0001, experiment 2 had the value set to 0.001 and experiment 3 used no weight decay. Mixup $\alpha = 0.8$, 0.3 and 0.9 where used in experiments 2, 3 and 4, respectively. Experiment 1 also had its hyper-parameters tuned with FC dim equal to 75, dropout rate of 0.7, convolutional dilatation of 2×1 and kernel sizes of 7×1, 5×1, 3×1 and 2×1 for each respectively layer.

Regarding **CCS**, the following same hyper-parameters were used for simple holdout and k-fold cross validation approaches. The choice of optimizers is the same for CSS, except experiment 2, which used AdamW. The initial learning rate were defined 0.01 for experiment 4 and 0.001 for the others. Weight decays of 0.1, 0.01, 1e-05 and 0 were used for experiments 1 to 4, respectively. Mixup α 0.9, 1.0 and 0.7 were used in experiment 2, 3 and 4. As in the CSS analysis, experiment 1 had some parameters adjusted, namely, FC dim 125, dropout rate 0.7, convolutional dilatation 2×1 and kernel sizes 2×1, 2×1, 2×1 and 5×1.

In the windowed holdout approach, we used Adam optimizer in all experiments. The learning rate was set to 0.01 in experiment 4 and 0.001 for the others. Experiments 1 and 4 had weight decay of 0.0001, experiment 2 had the value set to 0.001 and experiment 3 used no weight decay. Mixup α of 0.8, 0.3 and 0.9 where used in experiments 2, 3 and 4, respectively. Experiment 1 also had its hyper-parameters tuned, FC dim 75, dropout rate 0.7, convolutional dilatation 2×1 and kernel sizes 7×1, 5×1, 3×1 and 2×1. Weight decays were 0.01, 0.001, 1e-05 and 0 and experiments 1 to 4, respectively. Mixup $\alpha = 0.4$, 0.1, 0.4 were used for experiments 2 to 4. Finally, experiment 1 had extra hyper-parameters adjusted was FC dim 75, dropout rate 0.7, convolutional dilatation 2x1 and as kernel sizes 7×1, 5×1, 3×1 and 2×1.

4. Results and Discussion

Tables 1 and 2 presents the obtained results for CSS and CCS, respectively. For each approach, we selected the experiment with highest development UAR and used it for test evaluation. We also build two ensembles of ensembles (sum of probabilities), one including all experiments and other with the three highest development UARs.

Regarding **CSS**, CNN14 provided consistent results over all sampling methods for the development set, resulting in UARs

Table 1: *CSS experiments*

Exp.	Train		Devel		Test
	F1	UAR	F1	UAR	UAR
Baseline [14]	-	-	-	57.90	**72.10**
Simple Holdout					
SpiraNet	91.50	96.34	70.49	73.57	
CNN14	92.61	96.27	74.13	76.94	
ResNet-38	95.77	96.81	**76.40**	**78.39**	70.30
MobileNet	**97.95**	**99.38**	73.80	75.73	
Windowed Holdout					
SpiraNet	65.83	79.39	49.13	59.28	
CNN14	**90.32**	**95.93**	**76.25**	**76.08**	65.20
ResNet-38	84.56	90.86	72.65	75.00	
MobileNet	83.22	89.96	75.26	75.93	
Cross K-fold					
SpiraNet	(84.57)	(88.38)	(76.43)	(81.80)	
CNN14	**(99.82)**	**(99.90)**	**(87.37)**	**(90.59)**	68.90
ResNet-38	(99.65)	(99.81)	(86.34)	(89.75)	
MobileNet	(92.83)	(95.18)	(79.06)	(83.99)	
Heterogeneous Ensambles					
Baseline [14]	-	-	-	-	71.10
Top three	-	-	-	-	69.70
All Models	-	-	-	-	69.40

Table 2: *CCS experiments*

Exp.	Train		Devel		Test
	F1	UAR	F1	UAR	UAR
Baseline [14]	-	-	-	64.7	72.90
Simple Holdout					
SpiraNet	57.65	74.83	41.21	62.73	
CNN14	**91.61**	**96.97**	**51.42**	**69.92**	**75.90**
ResNet-38	85.02	94.18	47.76	68.57	
MobileNet	72.04	83.40	43.24	64.75	
Windowed Holdout					
SpiraNet	64.96	77.77	38.51	60.41	
CNN14	57.89	75.31	43.42	65.26	
ResNet-38	75.44	86.69	45.76	66.37	
MobileNet	**59.84**	**72.57**	**50.00**	**67.77**	68.90
Cross K-fold					
SpiraNet	(86.10)	(94.29)	(55.95)	(72.17)	
CNN14	**(99.79)**	**(99.93)**	**(76.79)**	**(86.60)**	69.60
ResNet-38	(99.79)	(99.93)	(70.75)	(82.48)	
MobileNet	(71.57)	(85.21)	(64.13)	(79.22)	
Heterogeneous Ensambles					
Baseline [14]	-	-	-	-	73.90
Top three	-	-	-	-	71.20
All Models	-	-	-	-	70.60

superior to 76%. In fact, this model lead to highest UARs in both windowed holdout and k-fold cross validation. The best UAR for simple holdout were obtained by the ResNet-38 model.

In [14], the authors reached a maximum UAR of 70.50% in the development set and this same model reached the second best result in the test with a UAR of 68.70%. Compared to this model, our best development model has a development UAR approximately 8% higher and a test UAR and 2% higher. However, a baseline with inferior development UAR leaded to the highest test UAR. Our model is approximately 2% inferior to this baseline. Additionally, our two heterogeneous ensembles had a similar result, but both failed to overcome the baseline in the test set.

The best model of the baseline in the test set was the second worst considering development set and jumped from 57.90% to 72.10% from one set to the other. This difference of approximately 14% may indicate that the development set is not a good representative of the test set. As it is a multi-language dataset, the unbalance of languages in the validation set can compromise learning, this information is unclear.

In **CCS** experiments, CNN14 also showed consistent results, presenting the highest UARs for simple holdout and k-fold cross validation sampling approaches and MobileNet presented the higher development UAR for the windowed sampling method.

In [14] the authors reached a maximum UAR of 66.40% in the development set and this same model reached a UAR of 67.60% in the test. Our best model in development shows superior results of approximately 3% when compared to the best model in the baseline development set.

In the test set, our best experiment using the simple holdout approach and the CNN14 architecture reached a UAR of 75.90%, which was 2% above the ensemble used as a baseline in the competition. In addition, we see baselines with superior performances in the test set compared to the performance in the development set. In [14], the best model was more than 8% better in test than in development. The same occurred with our best model it was approximately 6% better in test set. Finally, our two heterogeneous ensembles had results close to each other, however they did not surpass our best homogeneous ensemble performance.

The sampling methods showed a tendency to better results for simple holdout, followed by cross validation and windowed holdout, in this order. Despite cross validation presented promising results in related work [4], it was not able to surpass simple-holdout. One hypothesis is that this dataset is smaller to the one used in [4]. It is noticeable that k-fold was still superior to windowed holdout even without specific fine tunning as in the other two approaches.

5. Conclusions and future work

This paper presented a contribution for the CSS and CCS competitions by tackling the challenges using deep learning based models. Because such methods are data-hungry, in contrast with the size of the datasets available for the challenge, we explored both transfer learning and several data augmentation methods, in attempt to obtain competitive results. In this sense, we also explored instance sampling methods. Although our models could not overcome the baseline results for CSS, they were able to surpass them in CCS by 2%.

6. Acknowledgements

This work was supported by Fapesp project 2020/06443-5 (SPIRA). This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001. Marcelo Finger was partly supported by Fapesp projects 2019/07665-4 and 2014/12236-1 and CNPq grant PQ 303609/2018-4. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPU used in part of the experiments presented in this research.

7. References

[1] G. Chambres, P. Hanna, and M. Desainte-Catherine, "Automatic detection of patient with respiratory diseases using lung sound analysis," in *2018 International Conference on Content-Based Multimedia Indexing (CBMI)*. IEEE, 2018, pp. 1–6.

[2] D. Perna and A. Tagarelli, "Deep auscultation: Predicting respiratory anomalies and diseases via recurrent neural networks," in *2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*. IEEE, 2019, pp. 50–55.

[3] F. P. Dos Santos, C. Zor, J. Kittler, and M. A. Ponti, "Learning image features with fewer labels using a semi-supervised deep convolutional network," *Neural Networks*, vol. 132, pp. 131–143, 2020.

[4] J. Szep and S. Hariri, "Paralinguistic classification of mask wearing by image classifiers and fusion," *Proceedings INTERSPEECH. Shanghai, China: ISCA*, pp. 2087–2091, 2020.

[5] M. Markitantov, D. Dresvyanskiy, D. Mamontov, H. Kaya, W. Minker, and A. Karpov, "Ensembling end-to-end deep models for computational paralinguistics tasks: Compare 2020 mask and breathing sub-challenges," *INTERSPEECH, Shanghai, China*, 2020.

[6] T. Koike, K. Qian, B. W. Schuller, and Y. Yamamoto, "Learning higher representations from pre-trained deep models with data augmentation for the compare 2020 challenge mask task," *Proceedings INTERSPEECH. Shanghai, China: ISCA*, pp. 2047–2051, 2020.

[7] S.-L. Yeh, G.-Y. Chao, B.-H. Su, Y.-L. Huang, M.-H. Lin, Y.-C. Tsai, Y.-W. Tai, Z.-C. Lu, C.-Y. Chen, T.-M. Tai *et al.*, "Using attention networks and adversarial augmentation for styrian dialect continuous sleepiness and baby sound recognition." in *INTERSPEECH*, 2019, pp. 2398–2402.

[8] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[9] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[10] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *Proceedings INTERSPEECH 2019*, pp. 2613–2617, 2019.

[11] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[12] C. Brown, J. Chauhan, A. Grammenos, J. Han, A. Hasthanasombat, D. Spathis, T. Xia, P. Cicuta, and C. Mascolo, "Exploring automatic diagnosis of covid-19 from crowdsourced respiratory sound data," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3474–3484.

[13] J. Han, C. Brown, J. Chauhan, A. Grammenos, A. Hasthanasombat, D. Spathis, T. Xia, P. Cicuta, and C. Mascolo, "Exploring automatic covid-19 diagnosis via voice and symptoms from crowdsourced data," *arXiv preprint arXiv:2102.05225*, 2021.

[14] B. W. Schuller, A. Batliner, C. Bergler, C. Mascolo, J. Han, I. Lefter, H. Kaya, S. Amiriparian, A. Baird, L. Stappen, S. Ottl, M. Gerczuk, P. Tzirakis, C. Brown, J. Chauhan, A. Grammenos, A. Hasthanasombat, D. Spathis, T. Xia, P. Cicuta, M. R. Leon J. J. Zwerts, J. Treep, and C. Kaandorp, "The INTERSPEECH 2021 Computational Paralinguistics Challenge: COVID-19 Cough, COVID-19 Speech, Escalation & Primates," in *Proceedings INTERSPEECH 2021, 22nd Annual Conference of the International Speech Communication Association*. Brno, Czechia: ISCA, September 2021, to appear.

[15] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[16] H. S. Heo, B.-J. Lee, J. Huh, and J. S. Chung, "Clova baseline system for the voxceleb speaker recognition challenge 2020," *arXiv preprint arXiv:2009.14153*, 2020.

[17] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[18] S. Kornblith, J. Shlens, and Q. V. Le, "Do better imagenet models transfer better?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2661–2671.

[19] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *arXiv preprint arXiv:2006.11477*, 2020.

[20] M. Huh, P. Agrawal, and A. A. Efros, "What makes imagenet good for transfer learning?" *arXiv preprint arXiv:1608.08614*, 2016.

[21] F. P. dos Santos, L. S. Ribeiro, and M. A. Ponti, "Generalization of feature embeddings transferred from different video anomaly detection domains," *Journal of Visual Communication and Image Representation*, vol. 60, pp. 407–416, 2019.

[22] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 776–780.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[24] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[26] E. O. Brigham and R. Morrow, "The fast fourier transform," *IEEE spectrum*, vol. 4, no. 12, pp. 63–70, 1967.

[27] E. Casanova, L. Gris, A. Camargo, D. Silva, M. Gazzola, E. Sabino, A. Levin, A. Candido Jr, S. Aluisio, and M. Finger, "Deep learning against covid-19: Respiratory insufficiency detection in brazilian portuguese speech," in *Findings of the Association for Computational Linguistics: ACL 2021*. ACL, Aug. 2021, accepted for publication.

[28] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[30] T. Emara, H. M. Afify, F. H. Ismail, and A. E. Hassanien, "A modified inception-v4 for imbalanced skin cancer classification dataset," in *2019 14th International Conference on Computer Engineering and Systems (ICCES)*. IEEE, 2019, pp. 28–33.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[33] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *arXiv preprint arXiv:1908.03265*, 2019.

[34] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.