CNN Architectures for Mobile Device Cough Detection

Valentin Romero (varopr@edem.es), Ignacio Costa (nacona@edem.es), Ramon Marques (ramavi@edem.es), Felix Sanchez (fesato@edem.es) and German Valera (gevama@edem.es)

Abstract

Helped by the advances in Convolutional Neural Network (CNN) for image classification, we begin a study to prove its use for audio classification and recognition in architectures deployable in mobile devices. The event we want to primarily recognize is cough with the hopes to develop a way to track the frequency of it in the fight of COVID-19. In this study we use a variety of architectures and free access databases to prove that it can be achieved with a high degree of accuracy. We also discuss improvements to our model and case studies such as this.

1. Introduction

Image classification performance has improved greatly with the advent of large datasets such as ImageNet and the use of Convolutional Neural Network (CNN) architectures for instance DenseNet, VGG, Inception, and ResNet. Taking advantage of these performance improvements, we decide to see if CNNs deployable in mobile devices can yield good performance on audio classification problems. Our main goal is to detect “cough” among all the sounds on our datasets, and then develop a program that can detect such event in the background audio of a mobile device.

Our implementation was performed on Google Colab, but any GPU-enabled Python instance should be capable of achieving the same results. The data we use consists of the combination of two audio classifications datasets that make a total of 3769 audio files each tagged in 24 categories. We call this dataset DatasetMix. In this paper we explain the process we took in developing this CNN model based on constraints of having it to be deployable in a mobile device. To address this we based our decisions in size, validation accuracy, accuracy in recognizing cough sounds and the possibility of the model being used in an Android device.

For the purpose of finding our target model, this paper compares the performance of several different architectures using one of the main datasets (ESC-50). Once we find the architecture that best fits our case we will proceed with the training and evaluation of our model with the intention of achieving an accuracy high enough that validates our study. We will also validate the premise that Convolutional neural networks can classify sound clips to a high degree of accuracy through the use of image representations. Once we achieve said validation we will explore the deployment of said model within the purpose in which was requested, recognizing cough for an app to help fight the spread of COVID-19.

2. Dataset

During recent years many datasets have been developed to address the previously limited availability of labeled environmental sound data. One of these datasets is the Environmental Sound Classification (ESC) dataset, which was released by Piczak in 2015. For our case we use the ESC-50 dataset, which contains 2,000 5-second long audio clips equally divided among 50 categories. We then proceeded to filter the 50 categories down to 13 which we believe are closely related to sounds that a mobile device can be exposed to in our case study. The 13 sounds we decided to use of the dataset are knock, computer keyboard, cough, laughter, fart, telephone, bark, burping, meow, applause, finger snapping, shatter and drawer opening or closing.

Another dataset that we use for our study is the Freedsound Dataset Kaggle 2018 (or FSDKaggle2018 for short). This audio dataset contains 11,073 audio files annotated with 41 labels. This dataset was originally used for a challenge in the Detection and Classification of Acoustic Scenes and Events (DCASE) in 2018. In this dataset we repeated the process of selecting sound categories or labels that match our case study. In this case we selected the following 17 labels, dog, cat, pouring water, toilet flush, crying baby, sneezing, clapping, breathing, coughing, footsteps, laughing, brushing teeth, snoring, drinking sipping, door wood knock, can opening and glass breaking.

We proceeded to combine both datasets into one that gave us as a result 23 categories and 3,614 audio files, but due to the reason that we are going to predict based on the audio recorded in the background over long periods of time we added a 24th category named silence, bringing the total samples to 3,769. We did this knowing that we wanted to work with CNNs (which require a high amount of data) and wanted to have a high variety of sounds for the model to differentiate from our target. During our preprocessing stage we proceeded to separate the data in 80/20 Train and Test split for our following model training and evaluation.

3. Experimental set-up

3.1 Mel-Spectrograms

The next steps in our experimental process were to preprocess our data and to define our architecture. The preprocessing entails the transformation of our audio files into data that can be used by our model to make predictions. To do so we need to transform audio into images, and for that end we use Mel-Spectrograms. To understand why we use MFCC or Mel Frequency Cepstral Coefficients we need to dive into how spectrograms work. A typical spectrogram uses a linear frequency scaling, so each frequency bin is spaced in an equal number of Hertz. The Mel-frequency scale on the other hand, is a quasi-logarithmic spacing roughly resembling the resolution of the human auditory system. This makes the MFCC features more “biologically inspired”.

Our target Mel-spectrogram is generated using the Librosa package by applying a mel-filterbank to the magnitude spectrum of each segment, and then taking its logarithm.
In our set up we applied concurrent task so as we generate our spectrograms we also applied our data split for training and validation, giving us as a result our Mel-spectrograms saved in both training and test directories and ready to be processed by our model.

3.2 Architecture

From the beginning of our study we wanted to use Convolutional Neural Networks due to its state-of-the-art accuracy in image recognition and classification, the difference being that our purpose is audio classification. Having this difference meant that we needed to measure the different accuracies of the architectural models of CNNs. The other hurdle that we had in the building of our model was that it needed to be able to operate in an Android Mobile Device.

Given our case and the constraints we looked for models deployable in Tensor Flow Lite. TFL is the most recent open source deep learning framework for on-device inference. It works with both pre-trained supported models for a variety of uses including image classification and models that need to go through a conversion process to be deployable. After carefully comparing both Tensor Flow and TFL supported models, we ran basic tests in which we measured the accuracy of the models with a 10 category set of the ESC 50. For this comparison we used a 20 Epoch training using both the best models in Tensor Flow and Tensor Flow Lite.

Model	Size	Top-1 Validation Accuracy	Parameters	Depth	TF Lite
Xception	88 MB	0.6625	22,910,480	126	N
VGG16	528 MB	0.9125	138,357,544	23	N
ResNet101	171 MB	0.1875	44,707,176	-	Y
InceptionV3	92 MB	0.70	23,851,784	159	Y
MobileNetV2	14 MB	0.4125	3,538,984	88	Y
DenseNet201	80 MB	0.7125	20,242,984	201	Y
NASNetMobile	23 MB	0.25	5,326,716	-	Y

The results yielded that even when VGG16 had a better accuracy than all the models, its size and inability to properly convert to TFL made it an inviable option at this stage of experimentation. We proceeded then to use DenseNet architecture. DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters, which combined with its size and accuracy for our problem made it our target CNN for this proof of concept. Our final model was then built with a DenseNet 201 base architecture with the preset ImageNet weight and a sequential top model with the following specifications:

- Flatten Layer
- Dense Layer with 512 neurons and Relu activation function
- Dropout layer with a value of 0.5
- Dense Layer with 24 neurons and Softmax activation function as it is a multiclass classifier.

4. Results

After pre-processing our dataset and deciding on our architecture we proceeded to train our model with the 80% split of our dataset, using batches of 40 images and a 100 Epochs. The results were as follow:

Measurement	Accuracy
Validation Accuracy	0.8404
Top 5 Prediction	0.9617
Cough Prediction	0.79

We calculated Top 5 Prediction accuracy which means that the correct label is within the first five prediction of the model. This is because we can use this characteristic to further improve the performance of our model once we move into a deployable proof of concept. This improvement can come from a post-model processing that creates a set of rules, such as if the target sounds is the first 3 predictions mark it as a positive. The result of this means both an increase in accuracy and in false positives but given our case of detecting cough the benefit outweighs the cost.

5. Conclusions and Next Steps

We started this experiment trying to prove that CNNs deployable in mobile devices can work for audio classification specifically for the recognition of cough. By looking at our first result we can see that our findings are very promising as we achieved validation of our hypothesis and high degree of accuracy. We achieved this within the limitations of a Tensor Flow Lite supported model and a low amount of data specific to our case study. These limitations can also be looked as opportunities to further improve our study in the area.

The next step in our search to improve our base theory is to gather more data as it's related to our case. This means more audio files containing environmental sounds that a mobile device can be exposed and more cough sounds from a variety of users. This will greatly help us in further re-training of our model in the hopes of achieving higher accuracies. We will also like to develop a second model in which we can have a binary classification between cough related to COVID-19 (Dry Cough) and the other cases. This type of classification can work with the product of our first model in the way that it will filter cough from the rest of the other sounds to then determine if it´s related to COVID-19 or not. To accomplish this we will need a great amount of data of current patients with the virus to generate large enough dataset to train the model.

For a deployable proof of concept we will then have to develop a program that recognizes the events of cough as they happen in either recorded audio or live stream. To do so we will develop a windowing system that will allow us to create Mel-Spectrogram with a size similar to the ones we used for our training (2-5 sec of audio per window). By doing this we will be able to record the events of cough as they happen. This will allow us to create an end-to-end solution for a mobile device cough detection problem.

6. Acknowledgements

This being our first Deep Learning audio project the authors will like to thank Felix Fuentes and Jose Peris for the base knowledge and instruction.

7. References

- Shawn Hershey, *CNN Architectures for Large-Scale Audio Classification* arXiv:1609.09430 , 2017
- K. J. Piczak*. Esc: Dataset for environmental sound classification*, 2015.
- Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger *Densely Connected Convolutional Networks ,* arXiv:1608.06993 , 2018
- Eduardo Fonseca *,General-purpose Tagging of Freesound Audio with AudioSet Labels: Task Description, Dataset, and Baseline,* arXiv:1807.09902, 2018
- Brian McFee *librosa: Audio and Music Signal Analysis in Python,* Corpus ID: 33504 , 2015
- Benoit Jacob *,Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference,* arXiv:1712.05877, 2017