Urban Audio Plugin

Enrique Nueve

May 2020

1 Abstract

Onboard the nodes of SAGE, an audio recorder is attached. Through the captured audio, an array of tasks such as classification of sounds, filtering of noise, and noise-pollution forecasting could be performed. In this write-up, an initial way of creating an audio classifier is presented that takes into account the limitations of the edge and limited data samples available for training. For this initial task, I propose constructing an audio classifier for urban sounds. I use the UrbanSounds8k dataset, which consists of 10 different urban sounds, as an initial benchmark for creating an audio classifier for urban sounds. This report shows the performance of a few CNN models pre-trained on ImageNet and then used for transfer-learning over the UrbanSounds8k dataset. I also propose future work directions for audio-ml tasks on the edge. The code is located at (https://github.com/waggle-sensor/machinelearning/tree/master/UrbanAudioPlugin).

2 Data

As mentioned the UrbanSound8k dataset (https://urbansounddataset.weebly.com/urbansound8k.html) consists of ten urban sounds: air conditioner, car horn, children playing, dog barking, drilling, engine idling, gun shot, jackhammer, siren, and street music. Although the dataset can be downloaded from the main website, it can also be downloaded from the wget call shown below.

```
wget https://zenodo.org/record/1203745/files/UrbanSound8K.tar.gz –O urban8k.tgz
```

To pre-process the dataset, I follow the process documented in the paper [PSY20]. I first re-sampled the .wav files to 22.5 kHz. Then I converted the .wav files into three-channel MelSpectrograms using different window sizes and hop lengths of (25ms,10ms), (50ms,25ms), and (100ms,50ms) on each respective channel. The MelSpectrograms had an image size of 128 by 250, with three channels being an RGB image and was scaled between 0 and 1. Then the MelSpectrogram dataset, along with one-hot encoded labels of the target classes for each sound, was made into tfRecord files, given the models used were made with TensorFlow.

A second version of the dataset was made where augmentation was applied to the data. Following the previous process, the .wav files were still re-sampled to 22.5 kHz. Then new versions of each .wav file was made with the applications of one of the following data augmentation methods: background noise of urban sound 1, background noise of urban sound 2, background noise of urban sound 3, background noise of urban sound 4, pitch shift of -3.5, pitch shift of -2.5, pitch shift of -2, pitch shift of 1, pitch shift of 2, pitch shift of 2.5, pitch shift of 3.5, time stretch of .81, time stretch of .93, time stretch of 1.07, or a time stretch of 1.23. Then as before, the samples were made into three-channel MelSpectrogram with varying window and hop lengths and along with their one-hot labels made into tfRecord files.

3 Model

Inspired by [PSY20], I sought to use transfer learning to improve the quality of my audio classifier model. However, based on my search, I could not find a large pre-trained model for the task of audio classification. However, as reported by [PSY20], they were able to get rather strong performance with models pre-trained on ImageNet and then fine-tuned on audio datasets in the form of spectrograms. [PSY20] report achieving .8514 accuracy on UrbanSound8K with a DenseNet pre-trained on ImageNet. To verify the claims of [PSY20], I repeated their experiment with a DenseNet201 with initial weights from ImageNet. Given that we want to run this on the edge, I tried running the experiment with EfficientNetB4 [TL20], a model made through a neural architecture search that outperforms DenseNet201 slightly and uses millions fewer parameters.

4 Experiments and Results

Upon running the experiments, I found the results from [PSY20] to be false. It appears that the reported value of .8541 for DenseNet201 was not the average but the highest fold score for accuracy. Through my test, there is no way the reported score is possible. In regards to EfficientNetB4 vs DenseNet201, the performance was almost the same yet EfficientNetB4 uses fewer parameters. In regards to using the larger data augmented dataset, the performance (TBD).

Model	Epochs	Aug.	1	2	3	4	5	6	7	8	9	10	Mean Acc.
DenseNet201 (False Report [PSY20])	70	No	-	-	-	-	-	-	-	-	-	-	.8514
DenseNet201	5	No	.7709	.7263	.7351	.7737	.8525	.7788	.8066	.7406	.8639	.8124	.7861
EfficientNetB4	5	No	.7113	.8006	.7372	.7485	.8205	.7618	.7983	.7332	.8455	.8518	.7809
EfficientNetB4	8	No	.7342	.8153	.7167	.7697	.8205	.7618	.8317	.7295	.8517	.8614	.7893
EfficientNetB4	4	Yes	.7549	.8514	.7178	.7757	.8803	.7557	.7673	.7245	.8088	.8723	.7908

5 Future Work

5.1 Data

- Use scalograms based on continuous wavelet transformations. There is a package to do this called PyWavelets. Work such as [Cop+19] show improved performance using scalograms.
- Train a specific model on a large audio dataset such as AudioSet (https://research.google.com/audioset/) in order to do transfer learning.

5.2 Models

- Use a GAN or auto-encoder like model as a sound separator
- Rare sound identifier with an auto-encoder
- Try a method from https://wp.nyu.edu/sonyc/publications/

5.3 Task

- Speech separator
- Filter out traffic sounds
- Listen for rare events such as lightning hitting a tree or a branch snapping
- Identify animal sounds
- Identify different languages being spoken
- Analysis of noise of area over time
- Analysis of correlation of noise between locations
- Predict where and when a sound will be heard next based on previous noise

References

[Cop+19] Abigail Copiaco et al. "Scalogram Neural Network Activations with Machine Learning for Domestic Multi-channel Audio Classification". In: *2019 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. 2019, pp. 1–6. DOI: 10.1109/ISSPIT47144.2019.9001814.

[PSY20] Kamalesh Palanisamy, Dipika Singhania, and Angela Yao. *Rethinking CNN Models for Audio Classification*. 2020. arXiv: 2007.11154 [cs.CV].

[TL20] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].