Report

Dataset

The dataset used is the UrbanSound8K dataset, which is a collection 8,732 audio samples of urban sounds belonging to 10 classes -

`air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music.`

The duration of each audio file is 4 seconds or less. Along with the audio files, the dataset includes metadata such as class labels and a few other features.

	slice_file_name	fsID	start	end	salience	fold	classID	class	file
0	100032-3-0-0.wav	100032	0.0	0.317551	1	5	3	dog_bark	UrbanSound8K/audio/fold5/100032-3-0-0.wav
1	100263-2-0-117.wav	100263	58.5	62.500000	1	5	2	children_playing	UrbanSound8K/audio/fold5/100263-2-0-117.wav
2	100263-2-0-121.wav	100263	60.5	64.500000	1	5	2	children_playing	UrbanSound8K/audio/fold5/100263-2-0-121.wav
3	100263-2-0-126.wav	100263	63.0	67.000000	1	5	2	children_playing	UrbanSound8K/audio/fold5/100263-2-0-126.wav
4	100263-2-0-137.wav	100263	68.5	72.500000	1	5	2	children_playing	UrbanSound8K/audio/fold5/100263-2-0-137.wav

Features

For the task of audio classification, several features can be considered - such as temporal MFCC coefficients, spectral features, zero crossing rate, etc. But for the task of urban sound classification, which are generally complex and diverse, MFCCs and spectral features are more popularly used.

MFCCs are a representation of the short-term power spectrum of a sound sample. They are very compact and can capture the important aspects of human hearing perception. But they may not capture fine-grained spectral details well, which can be important in certain types of audio classification

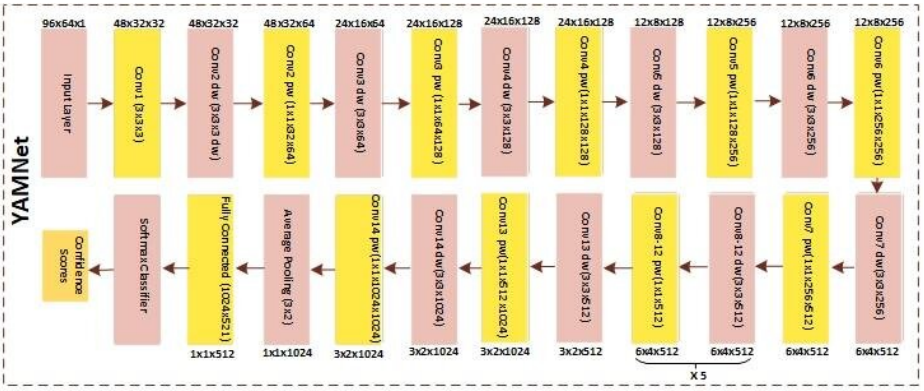
On the other hand, Mel spectrograms are created by computing the STFT of a sound signal, and mapping the frequencies to the Mel scale. They provide a time-frequency representation, which could potentially lead to a more robust classification. But the increased complexity, leads to higher computational complexity. They also might include redundant information for certain tasks, where a more compact representation like MFCCs would be enough.

Zero-shot Classification

Zero-shot classification refers to classifying data into categories that have been previously unseen by the model during the training process. This often involves the semantic relationships between classes. Here, the relation between the audio samples and the text (class descriptions) can be used to make new connections to unseen classes. To demonstrate zero-shot classification using , I have initially tried out the YAMNET model.

YAMNET

YAMNET or Yet Another Mobile Network is a deep learning model, which is designed for audio event classification and detection (*Drossos, Konstantinos, et al. "Sound event detection with depthwise separable and dilated convolutions." 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 2020.*). It is based on the architecture of MobileNetV1, which is a lightweight deep learning model designed for mobile applications. The architecture is CNN based, which ensures speed and memory efficiency, without sacrificing accuracy.



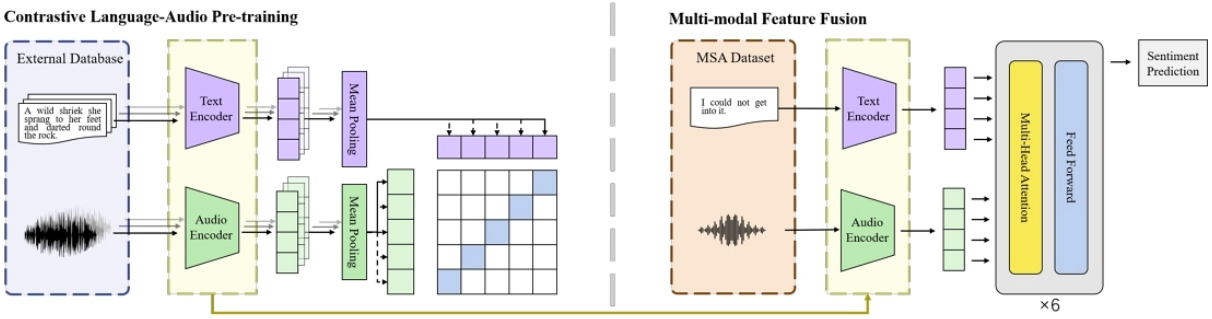
YAMNET architecture

The model is trained to recognise 521 audio events(classes), making it suitable for environmental sound classification. It takes short segments of audio and converts them into log-mel spectrograms, which are fed into the neural network. CNNs are great at extracting hierarchical spatial features from data, which implies that they are useful for analyzing the spectral features within a short time frame, like sound event detection or music genre classification from short clips.

I have used the pre-trained model to classify a few samples. Without fine-tuning the model, it maps the files into generalized classes, which might or might not capture the fine differences between the classes that are present in the dataset used. For example, an audio sample belonging to the class ‘dog_barking’ has been mapped to the class ‘Animal’, another sample belonging to the class ‘children_playing’ was mapped to the class ‘Speech’.

CLAP Model

In order to perform zero-shot classification, while taking the classes in the dataset into consideration, the model should be able to map audio data and text data into a shared embedding space, for a direct comparison. CLAP or Contrastive Language-Audio Pretraining utilizes a contrastive learning framework, which makes it able to distinguish between matching and non-matching pairs of audio and text data. (*Tianqi Zhao, Ming Kong, Tian Liang, Qiang Zhu, Kun Kuang, and Fei Wu. 2023. CLAP: Contrastive Language-Audio Pre-training Model for*
Multi-modal Sentiment Analysis. In International Conference on Multimedia Retrieval (ICMR '23)



CLAP Architecture

I used the pre-trained CLAP model to perform zero-shot classification of the audio samples, by providing it the set of labels as the text inputs. The model returns the probabilities for each label specified. For a lower number of classes, the model was highly accurate. The class with the highest confidence score can be taken as the predicted class.

For example, for an audio sample from the class ‘children_playing’, the model gave the same class a score of 0.9673, when there were only two classes given as the input.

```
[{'score': 0.9673612117767334, 'label': 'children playing'},
 {'score': 0.03263884410262108, 'label': 'dog bark'}]
```

When all the 10 classes were given -

```
[{'score': 0.6111621856689453, 'label': 'children playing'},
 {'score': 0.27071577310562134, 'label': 'street music'},
 {'score': 0.06318014860153198, 'label': 'siren'},
 {'score': 0.020620649680495262, 'label': 'dog bark'},
 {'score': 0.01248602569103241, 'label': 'gun shot'},
 {'score': 0.009232155047357082, 'label': 'car horn'},
 {'score': 0.007526170462369919, 'label': 'drilling'},
 {'score': 0.0023479561787098646, 'label': 'air conditioner'},
 {'score': 0.0019175066845491529, 'label': 'engine idling'},
 {'score': 0.0008114249212667346, 'label': 'jackhammer'}]
```

Training

For training, I opted for a simple neural network based architecture, with the MFCC coefficients of the audio samples as the main features for training to ensure efficiency and speed while not compromising the accuracy of the results. The model consists of 4 dense layers, 4 activation layers and 4 dropout layers.

After adjusting various parameters to optimize the training time and accuracy, the final model architecture is as follows -

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_16 (Dense)            (None, 600)               24600

 activation_16 (Activation)  (None, 600)               0

 dropout_12 (Dropout)        (None, 600)               0

 dense_17 (Dense)            (None, 600)               360600

 activation_17 (Activation)  (None, 600)               0

 dropout_13 (Dropout)        (None, 600)               0

 dense_18 (Dense)            (None, 600)               360600

 activation_18 (Activation)  (None, 600)               0

 dropout_14 (Dropout)        (None, 600)               0

 dense_19 (Dense)            (None, 10)                6010

 activation_19 (Activation)  (None, 10)                0

=================================================================
Total params: 751810 (2.87 MB)
Trainable params: 751810 (2.87 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Results and Analysis

Initially with just 10 minutes of training, 87.29% accuracy has been achieved. After slightly updating the parameters of the model, it achieved around 95.44% accuracy (~18 min train time).

Model	Test Accuracy	Test Loss	Recall	Precision	F1-Scores
classification.hdf5	`0.8729`	`0.4242`	`0.8139`	`0.9367`	[`0.95431477` `0.8915662` `0.75784755` `0.8042328` `0.92583114` `0.951049` `0.82666665`

					0.9358288 0.9457364 0.7229551]
classification_v1.hdf5	0.9066	0.3594	0.8654	0.9545	[0.9662338 0.9195403 0.83076924 0.8632708 0.9360613 0.9672897 0.86624205 0.94459105 0.93963253 0.81940705]
classification_v2.hdf5	0.9544	0.4250	0.8460	0.9572	[0.96875 0.9101797 0.7714884 0.84764546 0.9282051 0.9742389 0.81632656 0.9544235 0.96428573 0.8031914]

Fine-tuning CNN based models such as YAMNET might yield similar to higher accuracy, but due to using Mel spectrograms as the input features, the complexity involved would be much higher, and the processing, training and inference would take much longer and are computationally heavy.

We can see that zero-shot classification using CLAP, due to its multimodal approach, can classify the audio samples into expected classes. This requires no training which makes it fast and effective. But in the case of a high number of classes and classes that are semantically similar, the approach might lead to erroneous predictions.

Fine-tuning pre-trained models, or the simple neural net model trained on MFCC features ensures fast and accurate classification, making it a suitable choice for near real-time environmental sound classification.