Introduction

This research paper will be looking at the incorporation of distributed big data architecture and deep learning neural networks to solve for specific use cases. The problem area being examined as part of the practical work is classifying urban sounds for the purposes of reducing noise pollution. By not only monitoring noise levels, but being able to identify the sources of the noise pollution allows for the creation of plans and policies designed to combat a specific issue that has been uncovered, thereby improving quality of life for the local community. Neural networks are well-suited to audio classification for a number of reasons including robustness and flexibility, and so were chosen as the machine learning model to be used in the analysis. Neural networks are also scalable, which is important for urban sound classification, and to achieve a similar scalability for data storage, a distributed big data architecture was necessary. Apache Kafka is used throughout the analysis for this purpose.

In addition to the practical deployment there will be a literature review performed on key papers which examine the topic of neural networks and big data architecture as it relates to urban sound classification and storage. As a result of the practical research and the literature review, a critical evaluation of the key findings will be presented, implications for future research and current limitations will be discussed, as well as knowledge gaps which could be addressed with further research.

1.1 Topic Overview

“Urban Sounds are everyday audio events that occur daily, presenting unstructured characteristics containing different genres of noise and sounds unrelated to the sound event under study, making it a challenging problem” (Nogueira et al., 2022)

There are a number of reasons why people attempt to solve the challenging problem of urban audio recognition, ranging from the chosen topic area of noise pollution, to public safety for gunshot or vehicle crash alerts. City optimisation with the detection of traffic congestion allowing for the adjustment of traffic lights or rerouting of lanes. All of these use cases depend on a reliable and accurate model which can predict the audio data. It can also require a large amount of computational power and storage. For those two reasons a common methodology is to use a neural network model, a distributed big data architecture, and often both in combination.

Neural networks “name and structure are inspired by the human brain” (IBM, 2021), logical when considering they will be used for audio recognition, which according to Hennebert (1994) is a complex perceptive task which other machine learning algorithms struggle with. Once created the model will intake audio and classify it based on previous training data. These predictions are what would help solve or answer the use cases mentioned above.

Storage of data is also an issue and audio data is much larger than numerical or text based data, meaning that a big data storage solution will be required. Distributed storage systems allow for storing large amounts of data as compared to traditional databases by being “designed to scale horizontally across clusters of low-cost, moderate-performance servers” (Gorton and Klein, 2015)

1.2 Objective Statement

Using a large set of labelled audio data relating to urban sounds, create and train a neural network model which can be used to classify future audio. Use the trained model to classify audio, and by means of a distributed big data architecture, store the audio recording, and the prediction result, for later use.

1.3 Research Question

How have neural networks been used to classify urban sounds and are there predominant methods or models used? Has big data architecture been used in the storage of urban sound data and if so which types of architectures were implemented. Is there missing research which could help address the problem area?

2. Materials

The models created in this analysis were trained on an urban sounds audio database, containing over 8000 labelled sound excerpts from 10 different classes of sounds. This was downloaded via a zip file from the original website, “<https://urbansounddataset.weebly.com/urbansound8k.html>” and then unzipped. The unzipped folder contains a csv file with metadata including the class labels for the audio files. There are then 10 separate folders each with over 80 audio files. Using Python, a function is created to extract the mel frequency cepstral coefficients (MFCCs), which are “highly effective in audio recognition and in modelling the subjective pitch and frequency content of audio signals” (Xu et al., 2004). The input is a file path and metadata, which will be the urban sound recordings, while the output will be the MFCCs and the class label. The function is then used in a for loop to process all of the audio clips into a ‘feature\_list’, containing the MFCCs and a ‘label\_list’ holding the corresponding class label. Both are multi-dimensional numpy arrays. These are then saved to disk to avoid having to reprocess them all on subsequent analysis’.

3. State of the Art

3.1 Research Methodologies

3.2 Key Papers Reviewed

4. Methods

4.1 EDA

As part of the exploratory data analysis (EDA) basis information is gathered from the feature list, as well as the data frame created to represent the MFCCs and classes of each audio clip.After displaying the distribution of the class labels a t-test was performed to see which MFCCs had a statistically significant difference between the children playing and dog barking classes.

More visuals and statistical analysis is performed describing the dataset, and then anova tests are performed to list classes which do have statistically significant differences.

4.2 Sound Classification Model

A neural network model was decided for the project and it was implemented using the tensorflow and keras packages. For all the models the feature and label lists are converted to numpy arrays and the class labels are encoded. The labels are encoded as keras models cannot directly process categorical labels. The data is then split into training and testing sets with an 80/20 split.

The first model created used an input layer taking the feature vectors as inputs. The next layer is a dense layer with 128 neurons and the ‘relu’ activation function. The final layer is the output layer which has 10 neurons, one for each class and the ‘softmax’ activation functions. The model is compiled using the ‘adam’ optimiser, and then trained using the ‘fit.()’ method on the training data.

The next model uses a different structure but is still made up of input, output and hidden layers. In this case there are two dense layers, both using the ‘relu’ activation, with 256 and 128 neurons respectively. The model is compiled and fit in the same way as the first.

The final model was created using 10-fold cross-validation which splits the data into 10 folds and then trains on 9 and tests on 1, repeating the process for each fold. The model used is the same as the second with 2 dense layers. Each of the 10 models are compiled and fit in the same way as the first two models.

For all models classification reports are printed and Training and Validation accuracy line charts are visualised.

4.3 Kafka Topic Storage

After installing Apache Kafka on the machine and starting the Zookeeper and Kafka servers, Python code is run which records a predefined length of audio which is split into smaller chunks. These are then processed to extract the MPCC features which are then run through the saved neural network model in order to make a prediction. The audio clip and prediction are then sent to a Kafka topic as key pair values using the ‘KafkaProducer()’ function. Once the specified duration has been reached the process will terminate itself.

To verify that the process has worked, a prediction has been made and both it and the recording were successfully saved to the topic, another function, ‘KafkaConsumer()’ is used. This allows for the retrieval of messages from a topic. As the format of the stored data is known, the code includes functions for decoding the utf-8 encoded prediction string, as well as reading the audio back to a playable format from the bytes it was stored as.

5 . Literature Review

6 .Critical Analysis

6.1 Deep Learning Using Big Data

6.2 Big Data Architectures Incorporating Neural Networks.

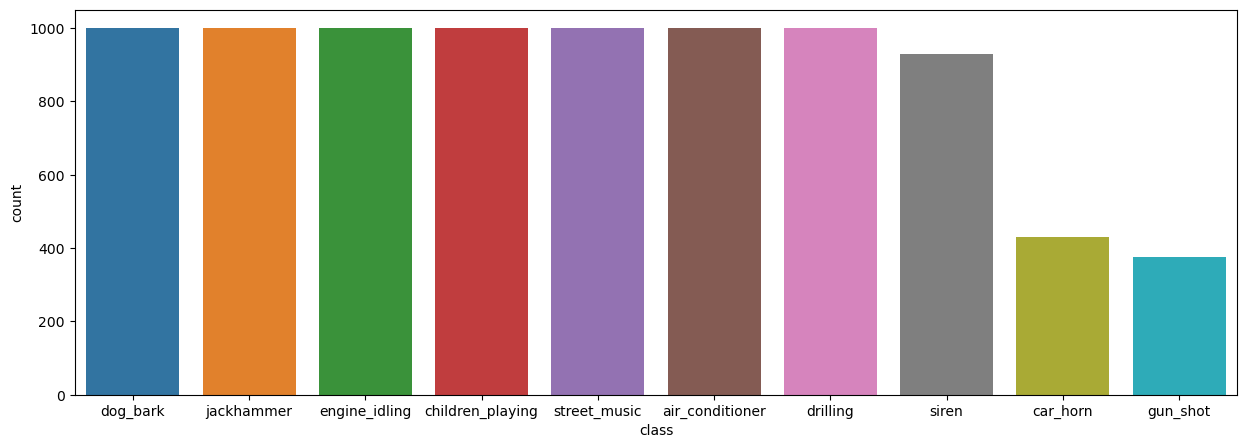
6.3 Implications and Limitations

6.4 Research Gaps

7. Results

7.1 EDA

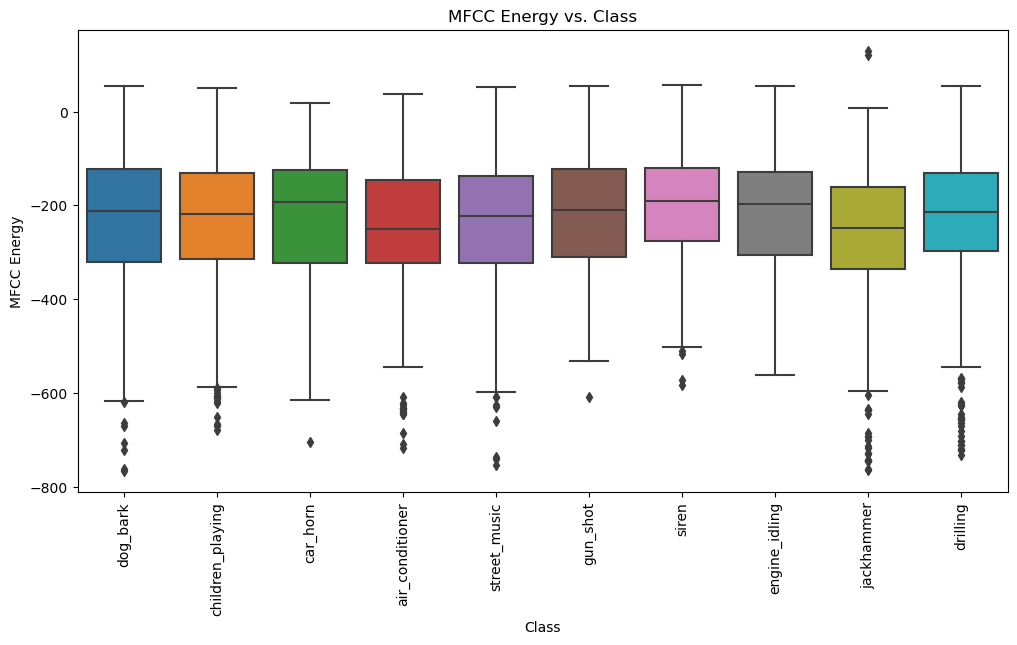
There are 8732 audio samples each with 13 different MFCC features. Examining the distribution of class labels shows 7 of the class labels have 1000 samples while ‘car\_horn’ and ‘gun\_shot’ have considerably less. This could affect the modelling so it will have to be taken into account.



The t-test showed that all features except for ‘mfcc\_spectral\_low’ had statistically significant differences in their means. These differences allow for the classification of different sound types.

The results of the ANOVA testing shows that all classes have differences in the means of at least one of its feature vectors. For most classes more than half of their features have statistically significant differences.

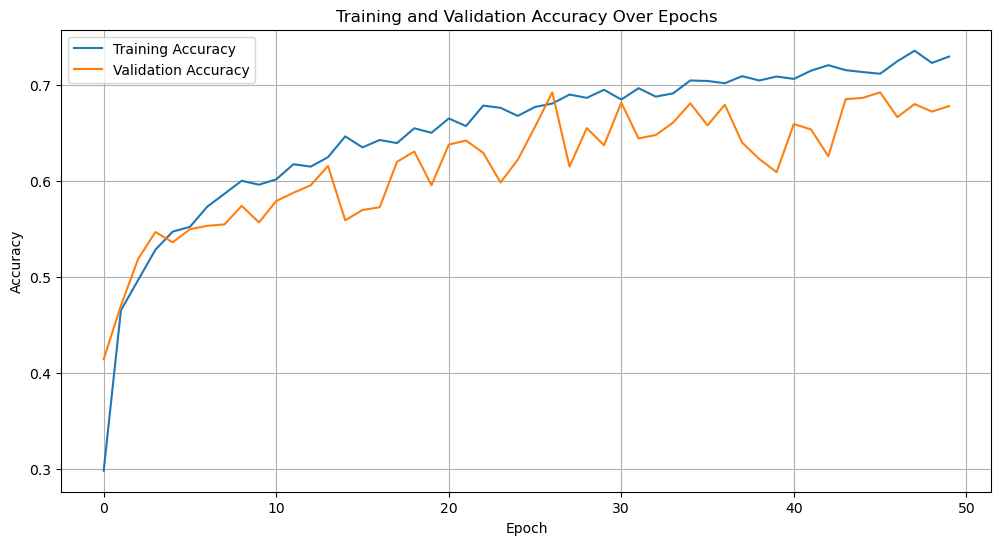
These variations can be seen when comparing a specific feature across classes, such as the ‘mfcc\_energy’ feature. Boxplots show that while some appear to have similar means, many classes vary significantly from one another.



7.2 Neural Network Model

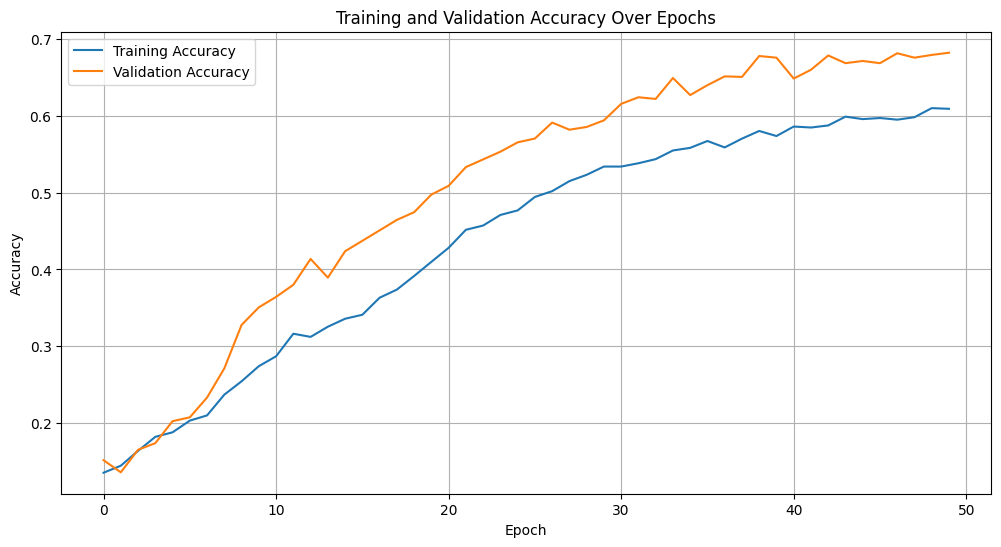
The first model returned a training loss of 0.76 with an accuracy of 0.74 on its final epoch.

The difference between the training and validation accuracy over time shows that the training accuracy is not significantly higher than the validation accuracy, which would be an indication of overfitting.

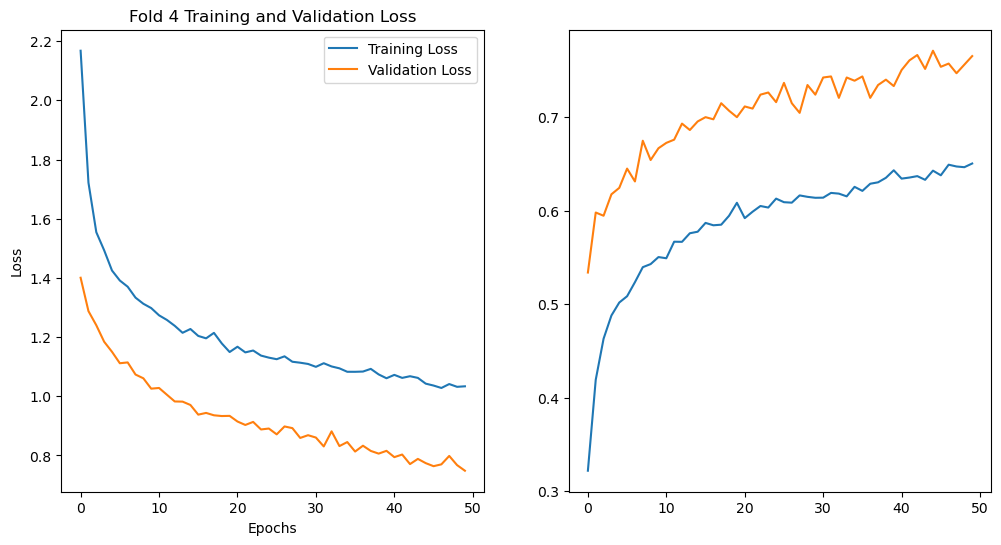


The classification report shows that ‘jackhammer’ and ‘siren’ are the classes with the highest f1-score, which balances precision and recall. These would be two loud and consistent noises so it is logical that they classify well as opposed to ‘street\_music’ for example which can be so varied that it is difficult to account for, resulting in low precision and recall results.

The second model results in training loss of 1.13 and an accuracy of 0.61 after its final epoch. This is a poorer performance than the first model. When comparing the validation results, validation loss is lower and accuracy is higher than the training results. This could indicate underfitting, however the differences are not extreme.



The third approach using 10-fold cross validation results in 10 separate models. Fold 4 has the best results with a training loss of 0.69 and a training accuracy of 0.78.



7.3 Kafka

The recordings and predictions were successfully sent to the topic, as verified by using kafka-consumer-console to view the messages. In terms of the results, during practical run throughs, when presented with sounds matching one of the class labels, the model has been able to classify each sound correctly and store this prediction as text to the kafka topic. The recording is also successfully stored, however on retrieval the audio is somewhat distorted. Listening to the audio before sending shows that it is clear and without distortion, so this occurs either when encoding the audio to be sent or decoding it on retrieval. As the model processing happens before encoding/sending the predictions are being made on clear audio, but this is an area where further work could significantly improve the quality.

8.1 Key Findings

8.2 Implications

8.3 Future Research

9. Conclusions