Computer Science
COC251
B525158

Classification of Environmental Sound Events

by

Callum I. Wright

Supervisor: Dr. A. Soltoggio

Department of Computer Science
Loughborough University

April/June 2019

Abstract

Environmental sound recognition is underdeveloped as a consequence of the scarcity of appropriate and openly available datasets. The release of ESC-50, UrbanSound8K and Google AudioSet have stimulated development of generalised audio research.

This paper explores supervised classification and unsupervised clustering techniques on AudioSet-15; a fifteen-class subset of AudioSet. The highest obtained mean AUC for a given model was 0.698. The highest obtained AUC for a single class was 0.875. Although poor model metrics were obtained, certain test audio samples were associated with correct predictions of very high probability. This provides a promising foundation for further development.

Contents

List of Abbreviations

ASA	Auditory Sound Awareness
AUC	Area Under Curve
CNN	Convolutional Neural Network
ELU	Exponential Linear Units
ESC	Environmental Sound Classification
kNN	k-Nearest Neighbour
LSTM	Long Short-Term Memory
MFCC	Mel-Frequency Cepstral Coefficient
MSE	Mean Square Error
NLL	Negative Log Likelihood
PReLU	Parametric Rectified Linear Unit
RMSprop	Root Mean Square Propagation
RReLU	Randomized Rectified Linear Unit
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SNE	Stochastic Neighbour Embedding
SVM	Support Vector Machine
TFRecord	TensorFlow Record
t-SNE	t-Distributed Stochastic Neighbour Embedding
VGG	Visual Geometry Group

Chapter 1

Introduction

1.1 Background

In recent years, machines capability to asses and interpret their environment has improved significantly, driven by advances in image recognition and physical spatial awareness. Nonetheless, these machines are commonly restricted by their lack of ASA; a fundamental component in constructing a complete understanding of an environment.

Image recognition is increasingly broad and varied, so too is audio. Sound provides a different perspective and information of an environment. Sound is not constrained by a linear perspective, travels through and around objects, and can provide insight that vision does not.

While the field of vision is well developed, the full potential of auditory recognition is underdeveloped as a consequence of underinvestment in datasets and appropriate research methods. Learning algorithms have no effective large scale and openly accessible datasets to train on, excluding speech recognition and music categorisation which represent just two subcategories of sound. Comprehensive datasets of all environmental sounds including those that are not audible to humans are not well developed and openly available. However, recent developments by Google AudioSet [1] are encouraging.

1.2 Aim and Objectives

1.2.1 Aim

The aim of this project is to advance environmental understanding of intelligent machines through improved ASA.

1.2.2 Objectives

There are three main objectives:

- To explore potential datasets and identify appropriate auditory learning techniques
- To produce a supervised classification or unsupervised clustering audio model
- To produce an outlier identification model

1.3 Applications

Conventional visual surveillance methods require a human operator to be present; one that is susceptible to attention fatigue. Surveillance is an integral component of security and the aid of an automated system would relieve an operator from such a tedious and mundane task. A more holistic audio based diagnostic system would be capable of identifying an event of interest that is not within view and one that is not reliant on optimal visual conditions e.g. lighting and adverse conditions.

Google AudioSet has constructed an extensive ontology of sound classes. Retraining the model(s) on a subclass of the ontology could result in further refinement of the hierarchy e.g. guitar brands, animal subspecies. This application may require additional relevant data.

1.4 Related Work

1.4.1 Datasets

In contrast to other areas in auditory recognition such as speech and music, there is limited research relating to environmental sounds due to a scarcity of appropriate datasets [2, pg. 1] [3, pg. 2]. The release of ESC-50 [4], UrbanSound8K [5] and Google AudioSet [1] have stimulated development of general audio research.

The ESC dataset was assembled to enrich the research domain in the hope to foster more open research in environmental sounds [4, pg. 4]. Following the release of the dataset, many papers were published regarding implementation of a variety of techniques trained on ESC and yielding a broad range of accuracies [6].

Different machine learning approaches may be favoured for the three subsets of the ESC dataset; ESC-50, ESC-10 and ESC-US. Both ESC-50 and ESC-10 consists of labelled samples and so are suitable for supervised learning; in contrast to ESC-US that consists of unlabelled samples, suitable for unsupervised learning. All samples were extracted from the Freesound project and are provided in a standardised format; five seconds of 44.1 kHz single channel audio. ESC-50 and ESC-10 aimed to minimise background noise and so samples were reviewed and hand-annotated. Consequently, a resource intensive process as such limited the volume of ESC-50 and ESC-10 to 2,000 and 400 samples respectively. ESC-US being unlabelled, consists of 250,000 samples.

UrbanSound8K consists of 8732 labelled samples up to four seconds that represent 10 urban sounds classes. Like ESC, the samples were extracted from the Freesound project. However, UrbanSound8K samples are not of a standardised format and may require further pre-processing.

Google AudioSet is described in chapter 3 and explored in chapter 4.

1.4.2 Audio Classification

With an abundance of research behind image classification using CNNs [7, 8], it is understandable that their techniques are adopted for use in audio classification. Several techniques representing audio data as an image have been implemented, albeit the predominant technique is spectrograms [2, 3, 9] and their adaptation to MFCCs [4, 10, 11, 12, 13].

Table 1.1 illustrates acclaimed models AlexNet and GoogleNet, and less acclaimed models WaveMsNet, L^3 and ConvRBM-BANK trained on ESC-50. The common denominator between the latter three models is the implementation of fused inputs. Both WaveMsNet and ConvRBM-BANK use raw waveform data as input, although differ in the way in which the data is represented. WaveMsNet simply fuses the waveform and log-mel features, whereas ConvRBM-BANK fuses Restricted Boltzmann Machines (RBM) with FilterBank Energies (FBE). L^3 harnesses the additional video recordings provided with ESC-50 to fuse two subnetworks; a visual subnetwork and an audio subnetwork. This technique is named Audio Visual Correspondence (AVC).

Table 1.1: ESC-50 Model Metrics.

Model	Accuracy
Piczak CNN [4]	64.50
AlexNet [14]	69.00
GoogleNet [14]	73.20
WaveMsNet after second phase [15]	79.10
L^3 [11]	79.30
ConvRBM-BANK [16]	86.50
Human performance [4]	81.30

Multiple papers [2, 3] note the insufficient size of available datasets, and suggest the use of data augmentation to increase the number of samples. Concerns of background noise [2, 10] and superimposed samples [17] have been highlighted and addressed by omitting samples [17].

1.4.3 Outlier Identification

Outliers in data are commonly identified as a by-product of unsupervised clustering techniques [18] e.g. SVMs, PCA and kNN. These techniques apply concepts of proximity which, in high dimensional space, fail to retain meaningfulness [18]. Many applications exist in high dimensional domains and so said techniques are often obsolete. These techniques are computationally expensive; worsening as the number of clusters, data points and data dimensions increase.

Multiple papers [18, 19, 20] suggest techniques that consider data points in a lower dimensional projection and identify an outlier as those data points that exists in a local region of low density. In comparison to a naive brute-force approach, the evolution-based approach [20] offers a small qualitative deduction in return of a large performance gain [18]. t-SNE [21] applies a similar method for dimensionality reduction to aid the visualisation of high

dimensional data.

Nonetheless, SVMs and kNN techniques have been explored for motor incident detection through anomalous sounds; determined as a vehicle crash or tyre skidding [10]. Low level audio features were extracted based on MFCC, energy ratios in Bark sub-bands and temporal-spectral characteristics. The SVM approach outperformed kNN for all feature sets in terms of recognition. In addition, result analysis exposed that "the kNN classifier [was] highly influenced by the loss in generalization capabilities when a high number of cluster[s are] configured." [10, pg. 287] The paper concludes that "features based on frequency-analysis (MFCC and BARK) have higher robustness to different SNR [Sound to Noise Ratio]." [10, pg. 287]

With an extensive collection of outlier identification techniques available [20], it is important to select the appropriate technique [20]. The effective performance of a technique is highly dependent on the composition of the dataset [18, 20]. SVMs are commonly chosen for binary data; labelled normal and abnormal. Although, pooled SVMs are able to handle high dimensional data [10], another approach was outlined; an evolution-based approach [18].

Chapter 2

Techniques

This chapter provides a description of techniques implemented in chapter 6 and is intended to clarify technical jargon. Those that are adept in the field of machine learning may find this chapter to be trivial. Following chapters reference sections within this chapter.

2.1 Supervised Learning

The following text will refer to classification as a supervised machine learning technique; inferring a function from labelled training data. Given a labelled dataset, a classification model aims to identify which set of categories or sub populations a new observation belongs to. This section will explore classification techniques relating to the classification of Google AudioSet.

2.1.1 Problem Definitions

Multi-Class

Multi-class classification refers to the assignment of a sample to one of more than two classes. This technique requires the sample labels to be mutually exclusive. Given n classes, a multi-class function h maps a sample X following $h : X \rightarrow \{0, \ldots, n\}$.

The baseline approach is to implement a One vs. All strategy; transforming the multi-class

classification problem into multiple binary classification problems. This provides a way to leverage extensive research and development into existing binary classification techniques. Using this technique, the number of classes is directly proportional to the number of models, thus reducing feasibility as the number of classes increases.

Neural networks provide a solution without the need to transform the initial problem through the implementation of various activation functions. These concepts are explored independently in section 2.1.2.

Multi-Label

Multi-label classification refers to the assignment of a sample to multiple classes. This task is a generalisation of the multiclass classification. Given n classes, a multi-label function h maps a sample X following $h : X \to \{0, 1\}^n$.

Time Series

Time series data refers to some measurements over a period of time. The meaning of elements in a time series are dependent on previous elements and reordering the series changes the meaning.

RNNs are an extension of a feed forward network. The defining concept of an RNN is the storage and use of information about a previous pass. As the model cycles through the series, the output is stored in memory within a hidden state. At each cycle, the hidden state is updated.

Given a sequence $x = (x_1, x_2, \cdots, x_r)$, the RNN updates the recurrent hidden state h_t following:

$$h_t = \begin{cases} 0, & t = 0 \\ \phi(h_{t-1}, x_t), & otherwise \end{cases} \tag{2.1}$$

where ϕ is a nonlinear function i.e. sigmoid with an affine transformation. [22]

Frequently, equation 2.1 is implemented following:

$$h^t = b + Wh^{t-1} + Ux^t \tag{2.2}$$

where the parameters are the bias vectors b and c along with the weight matrices U, V and W, respectively. [23, p. 374]

LSTMs are a type of RNN that provides a solution to the problem of degradation; a problem that RNNs suffer from. Degradation is the effecting outcome of *vanishing/exploding gradients*, explored in 2.1.5. LSTMs implement a memory cell that maintains information in memory for long periods of time. A set of gates are used to control the transfer of information: input gates, outputs gates and forget gates.

2.1.2 Activation Functions

The activation function of a neural network defines the output of a neuron. Given a series of inputs, an activation function maps the values to a desired output. Various activation functions exist, although ReLU, sigmoid and softmax see predominant use.

ReLU

The ReLU activation function is currently the most successful and widely-used activation function [24]. ReLU offers faster convergence [25] and superior performance [26]. Furthermore, ReLU is a non-saturated activation function that provides a solution to the *vanishing/exploding gradient* problem [25] yet suffers from the *dying ReLU* problem. ReLU computes the function $f(x) = max(0, x)$ as illustrated in figure 2.2a and defined as equation 2.3 [26]. It applies a threshold at zero and so activates only for positives values. During backpropagation and given a ReLU unit R, if an input to R is negative and has crossed the zero threshold, R will repeatedly output zero and can be said to have closed, thus a dead neuron. A closed ReLU cannot update its inputs parameters and so a dead ReLU neuron remains dead. This issue may be mitigated by a lower learning rate and the use of particular ReLU variants; Leaky ReLU [27], ELU [28] and RReLU.

$$y_i = \begin{cases} x_i & \text{if } x_i \geq 0 \\ 0 & \text{if } x_i < 0 \end{cases} \tag{2.3}$$

Figure 2.1b illustrates Leaky ReLU and the further variant PReLU [25]. These variants assign a linear non-zero gradient for negative values. Leaky ReLU represents a predefined gradient, whereas the PReLU gradient can be adjusted by the neural network; indicated by the dashed line in figure 2.1b. Leaky ReLU follows equation 2.4.

$$y_i = \begin{cases} x_i & \text{if } x_i \geq 0 \\ \frac{x_i}{a_i} & \text{if } x_i < 0 \end{cases} \tag{2.4}$$

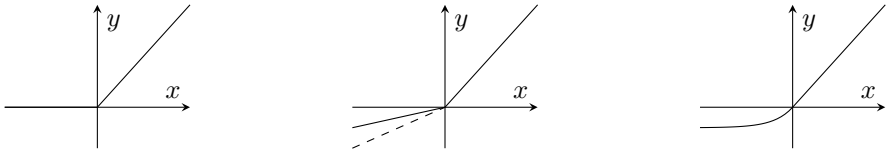
where a_i is a fixed parameter in range (1, $+\infty$) [26].

Figure 2.1c illustrates ELU. This variant assigns an exponential gradient for negative values. ELU implements a logarithmic curve and so saturates large negative values, reducing their negative impact.

RReLU assigns a random non-zero gradient for negative values [26]. Furthermore, RReLU has been reported to reduce overfitting due to its random nature [26, 29]. Given a random number a_{ji} sampled from a uniform distribution $U(l, u)$, RReLU follows equation 2.5.

$$y_{ji} = \begin{cases} x_{ji} & \text{if } x_{ji} \geq 0 \\ a_{ji}x_{ji} & \text{if } x_i < 0 \end{cases} \tag{2.5}$$

where a_{ji} $U(l, u) l < u$ and $l, u \in [0, 1)$ [26].



(a) ReLU (b) Leaky ReLU (c) ELU

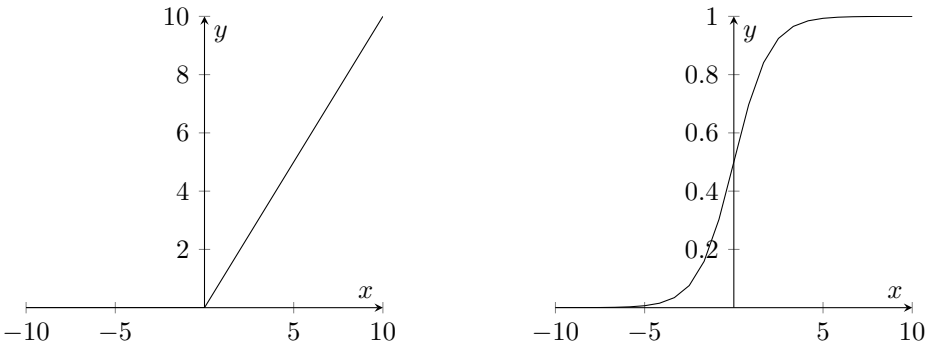
Figure 2.1: ReLU variants

Sigmoid

A sigmoid function may be implemented as the output layer of a neural network as a logistic function that produces a series of probabilities. It is differentiable and monotonic. The sigmoid function is best suited for binary classification. Sigmoid as illustrated in figure 2.2b, computes the function:

$$\frac{1}{1+e^{-x}}$$

Integrated throughout the model, a softmax function normalises n inputs into a probability distribution of n probabilities, where each probability lies within the range 0 to 1, and all probabilities sum equal to 1. Note that, the softmax function is a generalisation of the sigmoid function.



(a) ReLU (b) Sigmoid

Figure 2.2: Activation functions

2.1.3 Loss Functions

The loss function is a performance metric that represents the penalty of an incorrect prediction; the difference between the output and target. The robustness of the model increases as the value of the loss function decreases; thus the model aims to minimise loss.

NLL

Given n classes of correctly predicted probability values y_i, NLL is defined as equation 2.6. High predicted confidence correlates to a low loss value while a low predicted confidence correlated to a high loss value.

$$-\sum_{i=1}^{n} \log(\tilde{y}) \tag{2.6}$$

MSE

MSE is a statistical estimator that measures the mean squares of errors; the mean difference between the outputs and targets. Given n classes of true probability values y and predicted probability values $\tilde{y}$, MSE is defined as equation 2.7.

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \tag{2.7}$$

Cross-entropy

Given n classes of true probability values y and predicted probability values y_i, cross-entropy is defined as equation 2.8 and equation 2.9 for binary and non-binary classification respectively. Cross-entropy can be interpreted as the negative log likelihood of y_i under $\tilde{y_i}$.

$$-(y\log(\tilde{y}) + (1-y)\log(1-\tilde{y})) \tag{2.8}$$

$$-\sum_{i=1}^{n}(y_i\log(\tilde{y_i}) \tag{2.9}$$

2.1.4 Optimization Functions

Adam

Adam [30] is an adaptive learning rate optimization algorithm that was shown to yield increased performance in comparison to other optimization function. Despite a superior training time, the algorithm may not converge to an optimal solution and has been shown to generalize poorly [31].

Adam utilizes the benefits of AdaGrad and RMSprop; the ability to deal with sparse gradients and non-stationary objective [30]. The algorithm requires little memory and is computationally efficient, hence the superior training time.

SGD

SGD [32] is an iterative method for stochastic approximation of gradient descent; performing a parameter update for each training sample. The frequent updates with high variance result in a highly fluctuant metric. SGD with momentum β [33] approximately averages the gradient over $1/(1-\beta)$ updates.

RMSProp

RMSProp is an adaptive learning rate optimization algorithm developed to combat Adagrad's diminishing learning rates. RMSprop divides the learning rate by an exponentially decaying average of square gradients.

2.1.5 Common Issues

Vanishing/exploding Gradients

Vanishing gradients occur when gradients reduce in magnitude during backpropagation leading to a stagnation in learning. Exploding gradients occur when gradient errors accumulate and lead to large updates to the model weights during training. These problems results in accuracy saturation and then rapid degradation.

2.2 Unsupervised Learning

Clustering refers to the unsupervised learning technique; inferring a function from unlabelled data. Given an unlabelled dataset, a clustering model aims to group data points based on similarities so that the degree of association is maximised within each cluster and minimised outside of the cluster.

2.2.1 k-means

The k-means algorithm aims to partition observations into clusters such that the squared mean error between the empirical mean of a given cluster and the points in said cluster are

minimal [34].

Let $X = x_i, i = 1, ..., n$ be the set of n d-dimensional points to be clustered into a set of K cluster, $C = c_k, k = 1, ..., K$ and μ_k be the mean of cluster c_k. The squared error between μ_k and the points in cluster c_k is defined by [34] as:

$$\sum_{k=1}^{k} \sum_{x_i \in c_k} ||x_i - \mu_k||^2 \tag{2.10}$$

Equation 2.10 is applied through iterative refinements until a stopping criterion is met.

K-Means is guaranteed to converge to a result, although this may be a local minimum. Initialising random starting centroid is known to mitigate this issue.

2.2.2 t-SNE

SNE is a technique to visualise high-dimensional data; reducing data points to two or three dimensions. For each data point, a gaussian is centred in high dimensional space, under which in a lower dimension, a probability distribution is defined over potential neighbours [35]. The cost function aims to minimise the sum of kullback-leibler divergencies between the joint probabilities. Variations of SNE have been developed such as m-SNE [36] and t-SNE [21].

t-SNE is a variation of SNE that provides improved optimization and visualisation as data points are not clustered as centrally [21]. Further development of t-SNE has seen the implementation of the Barnes-Hut approximation [37], reducing the time complexity to O(NlogN) from the exact proposed algorithm of $O(N^N)$.

Chapter 3

Google AudioSet

3.1 Dataset

> AudioSet consists of an expanding ontology of 632 audio event classes and a collection of 2,084,320 human-labelled 10-second sound clips drawn from YouTube videos. The ontology is specified as a hierarchical graph of event categories, covering a wide range of human and animal sounds, musical instruments and genres, and common everyday environmental sounds. [38]

The dataset is available in two formats: three csv files describing each segment and a tar.gz containing 12,228 TFRecord files. The former three csv files provide a description of each segment but not a representation of the audio data. Downloading and extracting the data is not feasible in terms of time and so this option has been disregarded. The later provides a description of each segment in addition to 128-dimensional audio embeddings; extracted at 1Hz using a VGG-inspired model; VGGish. A TFRecord file is likely contain multiple records, where each record consists of the YouTube video ID, start time, end time, one or more labels and the 128-dimensional audio embeddings; where each feature is within the range of 0 to 255. The csv file *class_labels_indices* provided within the tar.gz maps integer indices to human readable class labels defined by the AudioSet ontology.

The dataset is pre-split into three subsets: balanced training; balanced validation; unbalanced training. Discrepancies are discussed in section 3.3. Dataset analysis is presented in chapter 4.

3.1.1 VGGish Model

VGG is commonly referred to as the CNN model produced by the Visual Geometry Group at the University of Oxford. Proposed in the paper [39], VGG, achieved 92.7% accuracy on ImageNet; a dataset containing 14 million images representing 1000 classes. VGG, being a deep CNN consisted of 19 layers; with a 16 layers alternate model commonly used.

VGGish is a variant of the VGG model. Changes made to the original VGG model:

- Input size to 96x64 for log mel spectrogram audio inputs.
- Final group of convolutional and maxpool layers removed.
- Replacement of final 1000-wide FC layer with a 128-wide FC layer.

[40]

The VGGish model applies the following, as specified from the Google AudioSet documentation [40]:

- All audio is resampled to 16 kHz mono.
- A spectrogram is computed using magnitudes of the Short-Time Fourier Transform with a window size of 25 ms, a window hop of 10 ms, and a periodic Hann window.
- A mel spectrogram is computed by mapping the spectrogram to 64 mel bins covering the range 125-7500 Hz.
- A stabilized log mel spectrogram is computed by applying log(mel-spectrum + 0.01) where the offset is used to avoid taking a logarithm of zero.
- These features are then framed into non-overlapping examples of 0.96 seconds, where each example covers 64 mel bands and 96 frames of 10 ms each.

3.2 Ontology

The ontology represents 632 categories as a hierarchical structure with a maximum depth of six levels. Released as a JSON file, each category consists of multiple fields: ID, Display name, Description, Examples, Children, Restrictions. The restrictions field applies to 78

categories; 56 blacklisted and 22 abstract. Blacklisted categories are not exposed to labellers as they are deemed obscure. Abstract categories acts as intermediate nodes to provide structure. The 632 categories are reduced to 554 with the exclusion of categories with flagged restrictions. [1]

3.3 AudioSet Discrepancy

Preliminary analysis on the AudioSet ontology and dataset revealed discrepancies between the publication and downloaded data.

The AudioSet ontology contained all 632 categories specified in the publication but consisted of an increased number of blacklisted and abstract categories; 67 blacklisted and 23 abstract in contrast to 56 and 22 respectively, specified in the publication. It is likely that this discrepancy is a result of outdated statistics within the publication in contrast to the revised ontology downloaded.

There are further discrepancies between the AudioSet publication and the downloaded dataset in terms of sample volume and label occurrence. Table 3.1 illustrates the volumetric discrepancy within the balanced training, evaluation and unbalanced training subsets, between the publication specification and the downloaded dataset.

The unbalanced subset downloaded contains a striking reduction in the number of samples; approximately half that the publication specified. Furthermore, there are discrepancies between the publication, the google website and the downloadable dataset regarding calculated volume as illustrated in table 3.1.

Table 3.1: Table of sample volume discrepancy between sources: publication [1], *research.google.com/audioset* and the AudioSet download. Calculated volume refers to the sum of balanced training, evaluation and unbalanced training.

Dataset Subset	Publication	research.google.com	Download
Balanced Training		22,176	18,973
Evaluation	17,748	20,383	18,644
Unbalanced Training	1,771,873	2,042,985	979,308
Stated Volume	1,789,621	2,084,320	
Calculated Volume	1,789,621	2,085,544	1,016,925

Chapter 4

Data Analysis

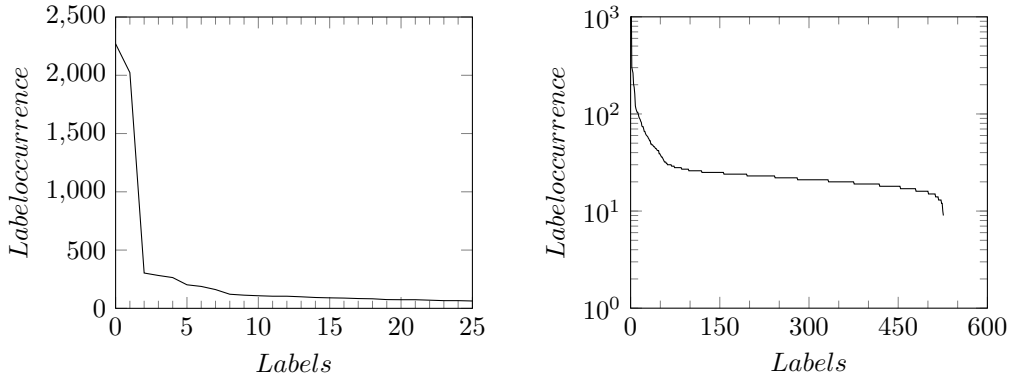
The decision to apply either a supervised classification or an unsupervised clustering approach is dependent on the data format and influenced by the quality of data. This chapter explores the analysis produced on the balanced training subset of the AudioSet dataset.

4.1 Label Distribution

Figure 4.1a highlights the significant range in label occurrence. Notably, *Music* and *Speech* represent 28% and 26% of the total label occurrence respectively. The heavy weighting towards these two labels produces a misleading mean value of 36, yet a skewness of 15 infers a distortion to the right; to the lower occurring labels. Regardless, this weighting will undoubtedly skew the model and reduce performance.

Figure 4.1b aids the visualisation of the mean, median and mode of the label occurrence; 36, 22 and 21 respectively. The lowest occurring labels are *Clapping* and *Ringtone* representing 0.053% and 0.048% respectively.

Further Statistical metrics are provided in table 4.1.



(a) Linear

(b) Logarithmic

Figure 4.1: Label occurrence. Labels are sorted in descending order of occurrence.

Table 4.1: Table of label occurrence statistics on the balanced training subset.

Statistic	Value
Mean	36
Median	22
Mode	21
Standard Deviation	133
Skewness	15
Minimum	9
Maximum	2270
Range	2261
Sum	18742

4.2 Feature Distribution

A boxplot is a standardized means of graphically representing a distribution of data based on five attributes: minimum, first quartile (Q1), median, third quartile (Q3) and maximum. For each class of the AudioSet ontology, a series of boxplots representing the distribution of feature values were produced. Further, more generalised statistics were generated from averaging these boxplot attributes.

Figure 4.2 and Figure 4.3 illustrate how the minimum and maximum feature values (FV) tend towards the median FV of 128 post feature index (FI) 96. This suggests that at the upper FI range, the FVs become more precise. This is supported by Figure 4.4 that illustrates the feature IQRs decreasing as the FIs increase. Over the 128 FI range, the IQR

value decreases by 50% from approximately 128 to 64.

Figure 4.5 illustrates that as the FI increases so do the number of outliers identified by the boxplots. As the graphs simply show the mean value of the individual classes, it is likely that the two heavily weighted labels *Music* and *Speech* increase the mean value of outliers.

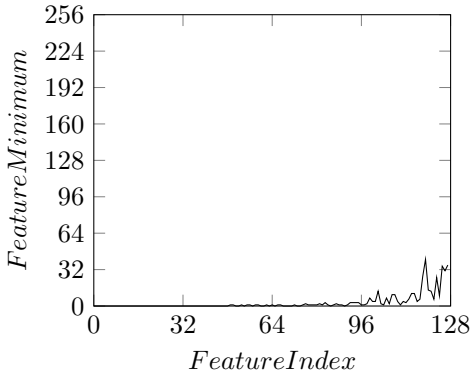


Figure 4.2: Quartile 1

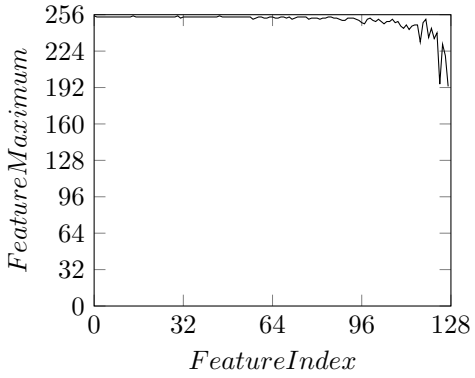


Figure 4.3: Quartile 3

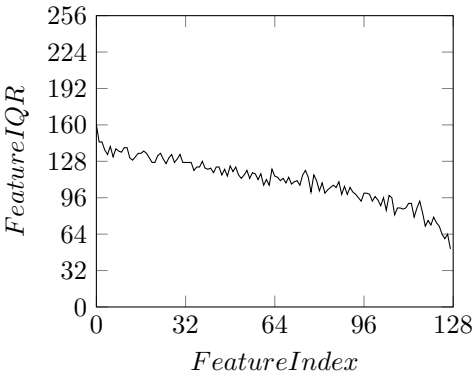


Figure 4.4: Interquartile Range

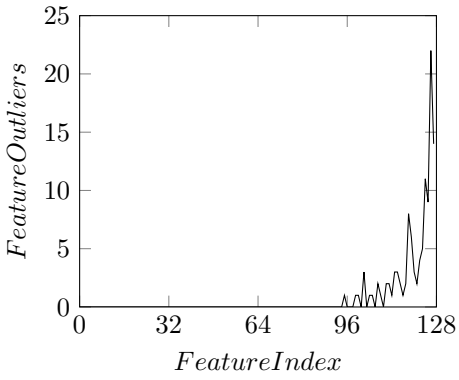


Figure 4.5: Number of Outliers

Chapter 5

Data Preparation

Data preparation is a necessary step of the machine learning process which often consumes substantial effort and time [41, p. 51]. The Google AudioSet required little preparation relating to the features but required considerable sample pruning.

5.1 Multiple Sample Labels

The AudioSet training samples are likely to be associated with multiple class labels; averaging 2.7 labels per sample [1]. This presents a problem for supervised classification as each training sample must be associated to a single class label [42]. Consequently, the decision was made to duplicate training samples and assign said duplicated samples to a single target label i.e. given a sample s assigned n labels l, n duplicates of s would be assigned a single unique label of l.

Duplicate samples that are associated to different target labels would likely adversely affect model performance. Thus, if a sample was to be added to the final dataset, the additional associated labels and their duplicate sample would not be i.e. given a duplicated sample s and its associated labels l_1, l_2 and l_3, if the association (s, l_1) were to be added to the dataset, associations (s, l_2) and (s, l_3) would not be.

5.2 Obtaining a Balanced Dataset

Section 4.1 highlighted the imbalance of label occurrence within the balanced training data and the need for a label occrence limit. Table 5.1 outlines the effect of capping the number of label occurrences on sample volume. The column *Obtained* refers to the number of samples extracted against the maximum number of samples obtainable. A lower value indicates a lesser balanced dataset. It was not until the limit was reduced to 3 that 100.00% was obtained. This indicated that there were classes with very low label occurrence. It is likely that the limited label occurence is a repercussion of omitting samples, as described in section 5.1. Consequently, the balanced and unbalanced training datasets were combined, resulting in the lowest label occurrence of 20. Nevertheless, there remained a trade-off between a balanced dataset and the volume of samples within the dataset. The final dataset consisted of a combination of the balanced and unbalanced training data. A label occurrence limit of 100 resulted in 54 classes falling short of of the 527 classes available.

Table 5.1: Table of limiting label occurrence on the unbalanced training subset. Maximum samples is the label limit multiplied by the number of unique labels; 527.

Limit	Maximum Samples	Unbalanced		Balanced and Unbalanced	
		Extracted Samples	Obtained	Extracted Samples	Obtained
25	13,175	13,039	98.98%	13,166	99.93%
50	26,350	25,599	97.15%	26,117	99.12%
75	39,525	37,821	95.69%	38,626	97.73%
100	52,700	49,553	94.03%	50,794	96.38%

5.3 Data Augmentation

Significant data augmentation was implemented by the VGGish model described in section 3.1.1 and so resulting features were simply normalised. Granted, the features were previously normalised between 0 and 255, the applied normalisation scaled the values to fall within a range of 0 and 1 as the transformation may increase the model's efficiency [43, p. 111].

5.4 Ontology Reduction

Initial models trained on all 527 labels yielded poor results. After exploring various model architectures and hyperparameter configurations that did not yield substantially better re-

sults, the decision was made to reduce the ontology. Two reduction methods were considered; filter by ontology level or filter by a predefined whitelist. The former was disregarded, as a single ontology level consisted of similar labels e.g. *Vehicle* and *Engine*.

The final dataset employed a whitelist that consisted of the following labels: Wind; Fire; Water; Domestic animals, pets; Livestock, farm animals, working animals; Wild animals; Musical instrument; Speech; Vehicle; Tools; Alarm; Explosion; Wood; Glass; Liquid. The label occurrence limit was set to 375; all labels fulfilling 375 occurrences. This dataset will be referred to as AudioSet-15.

Chapter 6

Models

6.1 Supervised Learning

6.1.1 CNN: Convolution Neural Network

Architecture

In this section, a *stack* is referred to as a generic structure of six layers: convolutional; dropout; convolutional; dropout; convolutional; maxpool. All convolutional layers apply a filter size of 3x3 with a padding of 1. An increased filter size impeded model learning and so was not implemented for further exploration; see appendix C. The number of output channels for a convolution is double that of the previous convolution. Two sequential convolutional layers that applied the same number of output channels yielded poor results. The dropout layer applies a probability of 0.5 as suggested by [44, p. 17]. The maxpool layer applies a kernel size and stride of 2.

The network input is a 10 x 128-dimensional list of normalised floating numbers that represent the audio embeddings produced by the VGGish model described in section 3.1.1. The network may be comprised of multiple *stacks* which are followed by two fully connected layers: the first with 1000 output channels and the second with 15; the number of classes. As a sample may represent multiple sounds and so map to multiple labels, the model attempts to solve a multi-label problem, thus a softmax layer is implemented.

Configurations

The network architecture is dependant of the configuration illustrated by table 6.1.

Table 6.2 outlines the output size of the final convolutional layer and the number of network parameters. A larger output size infers a smaller batch size given limited GPU memory capacity. A larger number of parameters infers a longer training time. The reduced output size and number of parameters for configuration C is the result of the secondary maxpool layer in combination with such a small number of channels of the following convolutional layer.

Table 6.1: Network configurations. Convolutional layers are of structure *conv<filter-size>-<number-of-channels>*.

<table>
<tr><td colspan="4">Network Configurations</td></tr>
<tr><td>A</td><td>B</td><td>C</td><td>D</td></tr>
<tr><td colspan="4">input (10 x 128)</td></tr>
<tr><td>conv3-16</td><td>conv3-32</td><td>conv3-32</td><td>conv3-32</td></tr>
<tr><td colspan="4">dropout</td></tr>
<tr><td>conv3-32</td><td>conv3-64</td><td>conv3-64</td><td>conv3-64</td></tr>
<tr><td colspan="4">dropout</td></tr>
<tr><td>conv3-64</td><td>conv3-128</td><td>conv3-128</td><td>conv3-128</td></tr>
<tr><td colspan="4">maxpool</td></tr>
<tr><td rowspan="6"></td><td rowspan="6"></td><td>conv3-32</td><td>conv3-256</td></tr>
<tr><td colspan="2">dropout</td></tr>
<tr><td>conv3-64</td><td>conv3-512</td></tr>
<tr><td colspan="2">dropout</td></tr>
<tr><td>conv3-128</td><td>conv3-1024</td></tr>
<tr><td colspan="4">maxpool</td></tr>
<tr><td colspan="4">FC-1000</td></tr>
<tr><td colspan="4">FC-15</td></tr>
<tr><td colspan="4">softmax</td></tr>
</table>

Table 6.2: Network configuration attributes.

Network configuration	A	B	C	D
Output size	20,480	40,960	8,192	65,536
Number of parameters (millions)	20.1	41.1	8.4	71.8

Early Stopping

Overfitting occurs when a model corresponds too closely to a particular set of data and has learnt the residual variance of the data i.e. noise. [45] As the residual variance present in the training data does not apply to new data, generalisation is reduced. A low training

error, yet a high validation error is likely to be a consequence of overfitting [46]. A higher validation error implies lesser generalisation [46].

Early stopping was implemented to prevent the model overfitting. The basic technique is to simply stop when validation error increases [46, 47]. In reality, this technique does not suffice as validation error may fluctuate, and as observed in various models, fluctuate rapidly initially. For this reason, early stopping is only applied post 10 epochs.

Furthermore, this basic technique is susceptible to local minima [47]. To minimise the likelihood of early stopping at a local minimum, a patience was defined i.e. the number of epochs to wait before stopping if validation error does not reduce. Nevertheless, there is a trade-off between training time and generalization [44]. Considering preliminary models, patience was set to 10.

6.1.2 LSTM: Long Short-Term Memory

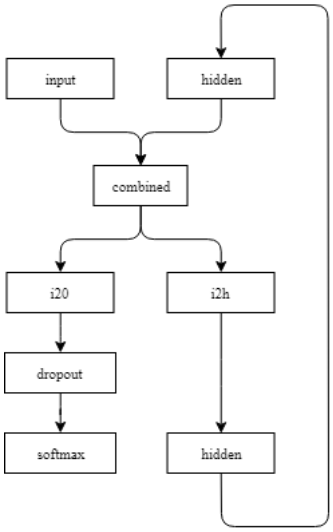
Although many LSTM variants exists, this section will focus on a vanilla LSTM; other variants include stacked, bi-directional, CNN and convolutional. LSTM models are a type of RNN model that implement feedback connection and so are suitable for time series data such as the AudioSet samples.

Architecture

The LSTM is fed the 10 x 128-dimensional list of normalised floating numbers that represent the audio embeddings produced by the VGGish model described in section 3.1.1. A single audio embedding is used as the input tensor and combined with a hidden tensor that is zero initialised. Illustrated in figure 6.1, these two tensors are combined and passed to the two sequential sets of layers; i2o and i2h. Although both i2o and i2h apply a linear transformation, i2o applies a further dropout and softmax computation. The output of the network is an output tensor and a hidden tensor. The hidden tensor is updated and fed to the network alongside the following audio embedding that substitutes the input tensor.

Various configurations were explored for the i2o and i2h layers, yet no configuration yielded superior results. Increasing the depth of the network yielded no improvements nor did increasing the width. The final configuration implemented a hidden size of 256. A dropout layer was implemented to reduce fluctuant metrics.

Figure 6.1: LSTM network.



6.2 Unsupervised Learning

6.2.1 t-SNE

t-SNE was implemented through the scikit-learn python project and fed 56,250 one-second 128-dimensional audio embeddings. Table 6.3 outlines the three parameters that were selected for alterations. The control value represents the default value provided by scikit-learn. Perplexity is related to the number of neighbours to be used in a single iteration. Larger datasets require a larger perplexity. Given that t-SNE is developed for use with millions of samples, this parameter should not yield substantially different results on a dataset of 56,250 samples. Furthermore, the scikit-learn documentation states that t-SNE is somewhat insensitive to this parameter.

Table 6.3: Parameter alterations. t-SNE is trained fifteen times. Each configuration will alter a single parameter and set the remaining two parameters to the control value.

Parameter	Control	Alterations
Perplexity	30	[5, 10, 30, 50, 75, 100]
Learning Rate	200	[100, 200, 250, 500, 750, 1000]
Number of Iterations	5000	[250, 500, 1000, 5000, 10,000]

Chapter 7

Results

7.1 Evaluation Data

AUC is a performance metric that represents the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance [48]. It considers both sensitivity and specificity and so provides a more comprehensive understanding of a model's performance, compared to accuracy and loss metrics. AUC requires predictions to be binary, thus the model predictions were translated into a binary representation post prediction.

The evaluation split of the AudioSet dataset was processed in a similar way to the training data described in chapter 5, although a label occurrence limit of 20 was applied. For each class, 20 true samples and a random selection of 20 false samples were predicted.

7.2 CNN: Convolutional Neural Network

7.2.1 Training Parameters

A learning rate of 0.001 yielded good results across various models during exploration and so was used for all models during training; see appendix B.

Adam, SGD and RMSprop referred to in section 2.1.4 were all explored for use as the

optimizer; see appendix A. SGD yielded better accuracy and loss, and was implemented with a momentum of 0.9.

Small batch training has been shown to yield greater generalization [49], thus a batch size was set to 256 and reduced by a factor of 2; see appendix D For all models trained with a batch size of 256, 128, 64, 32 and 16, as the batch size decreased, the AUC improved. The following text will refer to models trained with a batch size of 16.

7.2.2 Results

Table 7.1: Network configuration AUC for classes in AudioSet-15.

Class	Network Configuration			
	A	B	C	D
Speech	0.550	**0.600**	0.550	**0.600**
Domestic animals ...	**0.675**	**0.675**	**0.675**	0.650
Livestock ...	0.650	0.675	0.600	**0.700**
Wild animals	**0.825**	**0.825**	0.675	0.675
Musical instrument	**0.875**	0.850	0.850	0.750
Wind	**0.575**	0.550	**0.575**	0.525
Water	**0.775**	0.725	0.725	0.750
Fire	0.700	0.725	0.725	**0.775**
Vehicle	0.725	0.725	**0.825**	**0.825**
Alarm	0.700	0.700	0.650	**0.750**
Tools	**0.575**	**0.575**	0.550	**0.575**
Explosion	**0.775**	0.725	0.625	0.675
Wood	0.625	0.550	**0.650**	**0.650**
Glass	0.750	0.700	0.725	**0.775**
Liquid	0.700	0.575	**0.775**	0.750
Mean	**0.698**	0.678	0.678	0.695

All configurations converged at an accuracy below 50% with configurations A, B, C and D achieving 46.4%, 41.6%, 40.0% and 43.7% respectively. Table 7.1 illustrates the marginal difference in performance between the configurations to the extent that no clear distinctions can be drawn. With reference to AUC metrics, no configuration yielded superior metrics.

Nevertheless, table 7.1 does reveal potential issues with the data. Labels *Speech*, *Wind* and *Tools* are associated with poor AUC metrics for all four configurations. This is likely to be due to noise within the data samples i.e. those labels are likely to be present in other samples. Plotting confusion matrices revealed understandable misclassifications. *Wind* was frequently misclassified as *Vehicle*. *Tools* was frequently misclassified as *Wood*.

An AUC score of 1 infers a classifier that predicts correctly 100% of the time, while an AUC

score of 0 infers a classifier that predicts incorrectly 100% of the time. There was no case where a model yielded an AUC score less that 0.500; inferring no model predicts incorrectly more than it does correctly for any given class.

7.3 LSTM: Long Short-Term Memory

7.3.1 Training Parameters

A learning rate of 0.001 yielded marginally better metrics in comparison to 0.0001, whereas a learning rate of 0.01 hindered the model's ability to learn. NLLLoss was implemented as the loss function. The combination of NLLLoss and LogSoftmax as CrossEntropyLoss did not improve metrics.

7.3.2 Results

All configurations struggled to increase accuracy yet managed to decrease loss. Over the 100,000 sample limit, loss plateaued at approximately 70,000 samples.

Like the CNN described in section 7.2, labels *Speech*, *Wind* and *Tools* are associated with poor AUC metrics for the LSTM model.

Table 7.2: AUC for classes in AudioSet-15.

Class	AUC
Speech	0.575
Domestic animals ...	0.700
Livestock ...	0.675
Wild animals	0.750
Musical instrument	0.875
Wind	0.550
Water	0.625
Fire	0.700
Vehicle	0.725
Alarm	0.725
Tools	0.600
Explosion	0.600
Wood	0.675
Glass	0.775
Liquid	0.675
Mean	0.682

7.4 Acclaimed Models

Both VGG and ResNet failed to learn, predicting a single class repeatedly. In an attempt to solve this issue, dropout layers with a probability of 0.5 were added between various layers. Nevertheless, no improvements were seen.

7.5 Unsupervised Learning

The visualisation of the t-SNE technique applied to the data did not appear to elicit any clear data clusters. In both the two-dimensional and three-dimensional plots, the data points appeared random. A low iteration value and perplexity too, produced less sparsely distributed data points.

Chapter 8

Implementation

8.1 Process

A terminal based script was written to implement the models and provide a basis for further development and integration. The aim was to provide a simplified means of viewing the model predictions through a convenient and accessible application. A web interface was considered although disregarded as the implementation would add unnecessary dependencies and complexity. The terminal application requires Python and associated modules indicated by the *requirements.txt* file.

Audio data is either provided by the user or captured from the default microphone using the PyAudio module. The VGGish predictor that produces the 128-dimensional audio embeddings is robust and will format the data as necessary. Nevertheless, captured audio is of single channel 44.1kHz frames.

Given a file and a model, the application will produce VGGish embeddings and time splice according to the given model; ten seconds and a one second for the CNN and LSTM models respectively. The top three predictions and their score are displayed for each time splice. Additionally, the prediction scores are averaged and the top three are displayed.

8.2 Observations

The performance of the models is reflective of that described in chapter 7. Although the CNN model did not perform well, the LSTM performed significantly worse.

Six test files were compiled and fed to both the CNN and LSTM model. All files were between 120 and 240 seconds. The prediction and probabilities are outlined in table 8.1 and table 8.2. The test files are as follows with likely prediction italicised:

- Action scene from a film. *Explosion, Speech*
- Heavy diesel engine. *Vehicle*
- Guitar. *Musical Instrument*
- Documentary of a bear hunting fish in a river. *Water, Liquid, Speech, Wild Animals*
- Speech highlights of Barack Obama. *Speech*
- Wild animal samples. *Wild Animals, Domestic Animals*

The CNN model performed well for both the *Engine* and *Guitar* samples with probabilities 0.947 and 1.000 respectively. These samples did not contain conflicting sounds which is likely to be the reason for the high probabilities. Incorrect predictions of *Nature* and *Speech* are reflective of the poor AUC scores illustrated in table 7.1. Both samples include much speech which may be a sign that the label *Speech* is misclassified as *Wind*. The class *Fire* may be predicted in periods of no sound and general white noise which is prevalent in the *Wild Animals* and *Speech* samples. Interestingly, the label *Wild Animals* was not among the top three predictions for the sample *Wild Animals* and instead the labels *Domestic Animals* and *Livestock* although similar, was predicted with higher probability.

The LSTM performed poorly and appears to be heavily skewed towards *Fire*, often predicting the class in cases with no relevance. Although the sample *Engine* was predicted correctly with a probability of 0.441, it was closely followed by *Speech* with a probability of 0.388. The sample does not contain speech and so this sample prediction is dubious. The *Action Scene* sample does contain speech but not to the extent predicted; 0.323. Again, like the *Engine* sample, it is closely followed by an incorrect and irrelevant prediction; *Liquid* with a probability of 0.315.

Table 8.1: Predictions and probabilities of the CNN in configuration A for given audio sequences.

Audio	Top three predictions					
	Prediction 1		Prediction 2		Prediction 3	
Action Scene	0.227	Explosion	0.227	Livestock	0.136	Fire
Engine	0.947	Vehicle	0.053	Wind	0.000	Fire
Guitar	1.000	Musical Instrument	0.000	Liquid	0.000	Glass
Nature	0.737	Wind	0.158	Fire	0.053	Wild Animals
Speech	0.715	Wind	0.226	Fire	0.059	Speech
Wild Animals	0.533	Fire	0.200	Domestic Animals	0.133	Livestock

Table 8.2: Predictions and probabilities of the LSTM for given audio sequences.

Audio	Top three predictions					
	Prediction 1		Prediction 2		Prediction 3	
Action Scene	0.323	Speech	0.315	Liquid	0.281	Fire
Engine	0.441	Vehicle	0.388	Speech	0.114	Liquid
Guitar	0.287	Fire	0.265	Speech	0.242	Liquid
Nature	0.626	Fire	0.151	Liquid	0.092	Wind
Speech	0.548	Fire	0.210	Liquid	0.078	Speech
Wild Animals	0.273	Fire	0.183	Tools	0.183	Speech

Chapter 9

Conclusion

9.1 Summary

Section 1.4 highlighted the lack of appropriate datasets for generalised audio classification. High quality datasets are the foundation of research and development i.e. the CIFAR dataset may have been the catalyst for extensive research on image classification techniques. Moreover, limited research impedes development of insight and applications that may benefit society.

AudioSet is the most extensive dataset available that consists of environmental sounds. This greatly aids the development of generalised models capable of identifying a diverse variety of sounds. Although a few discrepancies exist as described in section 3.3, necessary pre-processing will resolve those issues.

Most dataset ontologies are flat and contain less than 100 classes. The AudioSet ontology is thorough and provides a hierarchical structure consisting of over 527 classes; including flagged classes. This caters for the needs of specific and generalised techniques.

Google AudioSet is a versatile dataset and ontology. Although raw audio and visual data is not included in the downloadable dataset, the enclosed files provide a reference to a means to obtain said data; albeit it is somewhat impractical to download all the referenced data. Nevertheless, a dataset that is authored and devloped by an influential organisation is promising for the field.

In practice, various techniques were explored to not yield acceptable results i.e. an acceptable AUC exceeding 0.750. No single model yielded superior metrics worthy of further investigate and more sophisticated deeper models yielding marginal gains and losses. Furthermore, acclaimed models described in section 7.4 failed to learn. The poor performance of the unsupervised technique t-SNE was an indication that other unsupervised techniques may perform poorly, and so further exploration of unsupervised techniques ceased.

Poor model metrics are attributed to issues with the pre-processing of data. The predominant issue discussed throughout this paper is how to manage multiple labels that are assigned to a single sample. Although the poor model metrics were reflected throughout implementation, certain samples were associated with correct predictions of very high probability, providing a promising foundation for further development.

9.2 Future Developments

To advance the existing models, the primary focus is undoubtably finding an appropriate technique to address the issue of multiple labels assigned to a single sample. This issue underpins the success of possible applications described in section 1.3.

References

[1] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audioset: An ontology and human-labelled dataset for audio events," report, Google, 2017.

[2] J. Salamon and J. P. Bello, "Deep convolutional neural networks and dataaugmentation for environmental sound classification," report, 2016.

[3] K. J. Piczak, "Environmental sound classification with convolutional neural networks," report, Institute of Electronics Systems and Warsaw University of Technology, 2015.

[4] K. J. Piczak, "Esc dataset for environmental sound classifiation," report, Institute of Electronics Systems and Warsaw University of Technology, 2015.

[5] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," report, New York University, 2014.

[6] karoldvl, "Esc-50," 2015.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," report, University of Toronto, 2012.

[8] D. Ciresan, U. Meier, and J. Schmidhuber, "Mulit-column deep neural networks for image classification," report, IDSIA and USI-SUPSI, 2012.

[9] A. Kumar, M. Khadkevich, and C. Fugen, "Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes," report, Carnegie Mellon University and Facebook, 2018.

[10] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Audio surveillance of roads: A system for detection anomalous sounds," report, 2016.

[11] R. Arandjelovic and A. Zisserman, "Look, listen and learn," report, DeepMind, VGG, 2017.

[12] D. M. Agrawal, H. B. Sailor, M. H. Soni, and H. A. Oatil, "Novel teo-based gammatone features for environmental sound classification," report, Dhirubhai Ambani Institute of Information and Communication Technology, 2017.

[13] M. Huzaifah, "Comparison of time-frequency representations for environmental sound classification using convolutional neural networks," report, Unknown, 2017.

[14] V. Boddapati, A. Patef, J. Rasmusson, and L. Lundberg, "Classifying environmental sounds using image recognition networks," report, Unknown, 2017.

[15] B. Zhu, C. Wang, F. Liu, J. Lei, Z. Lu, and Y. Peng, "Learning environmental sounds with multi-scale convolutional neural network," report, National University of Defense Technology, 2018.

[16] H. B. Sailor, D. M. Agrawal, and H. A. Patil, "Unsupervised filterbank learning using convolutional restricted boltzmann machine for environmental sound classification," report, Dhirubhai Ambani Institute of Information and Communication Technology, 2017.

[17] N. Sawhney, "Situational awareness from environmental sounds," report, Speech Interface Group and MIT Media Lab, 1997.

[18] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," report, IBM, 2002.

[19] C. Faloutsos, F. Korn, A. Labrinidis, Y. Kotidis, A. Kaplunovich, and P. D., "Quantifiable data mining using principal component analysis," report, Institute for Systems Research, 2001.

[20] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," survey, White Rose University Consortium, 2004.

[21] L. V. D. Maaten and G. Hinton, "Visualizing data using t-sne," report, Tilburg University and University of Toronto, 2008.

[22] J. Chung, Culcechre, K. Cho, and Y. Bengio, "Empirical evaluation of gated reccurent neural networks on sequence modeling," report, University de Montreal, 2014.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[24] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," report, Google, 2017.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deeper into rectifiers surpassing human-level performance on imagenet classification," report, Microsoft Research, Unknown.

[26] B. Xu, N. WAng, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolution network," report, University of Alberta and Hong Kong University of Science and Technology and University of Washington and Carnegie Mellon University, Unknown.

[27] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," report, Standford University, Unknown.

[28] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units," report, Institute of Bioinformatics, 2016.

[29] Kaggle, "National data science bowl," 2015.

[30] D. P. Kingma and J. L. Ba, "Adam a method for stochastic optimization," report, OpenAI and University of Toronto, 2014.

[31] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," report, University of California and Toyota Technological Institute, 2018.

[32] H. Robbins and S. Monro, "A stochastic approximation method," report, University of North Carolina, 1951.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back propagating errors," report, Nature, 1986.

[34] A. K. Jain, "Data clustering 50 years beyond k-means," report, Michigan State University and Korea University, 2010.

[35] G. Hinton and S. Roweis, "Stochastic neighbor embedding," report, University of Toronto, Unknown.

[36] B. Xie, Y. Mu, D. Tao, and K. Huang, "m-sne multiview stochastic neighbor embedding," report, Unknown, 2011.

[37] L. V. D. Maaten, "Accelerating t-sne using tree-based algorithms," report, Delft University of Technology, 2014.

[38] Google, *Google AudioSet*, 2019 (accessed January 2, 2019). `https://research.google.com/audioset/index.html`.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large scale image recognition," report, University of Oxford, 2015.

[40] Google, *Models for AudioSet: A Large Scale Dataset of Audio Events*, 2018 (accessed January 2, 2019). `https://github.com/tensorflow/models/tree/master/research/audioset`.

[41] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining Practial Machine Learning Tools and Technique*. Morgan Kaufmann, 2011.

[42] R. Jin and Ghahramani, "Learning with multiple labels," report, Carnegie Mellon University and University College London, Unknown.

[43] J. Han, M. Kamber, and J. Pei, *Data Mining Concept and Techniques*. Morgan Kaufmann, 2012.

[44] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout a simple way to prevent neural networks from overfitting," report, University of Karlsruhe, 2014.

[45] K. P. Burnham and D. R. Anderson, "Multimodel inference," report, Colorado Cooperative Dsh and Wildlife Research Unit, 2004.

[46] L. Prechelt, "Automatic early sopping using cross validation quantifying the criteria," report, University of Karlsruhe, 1997.

[47] L. Prechelt, "Early stopping - but when?," report, University Karlsruhe, Unknown.

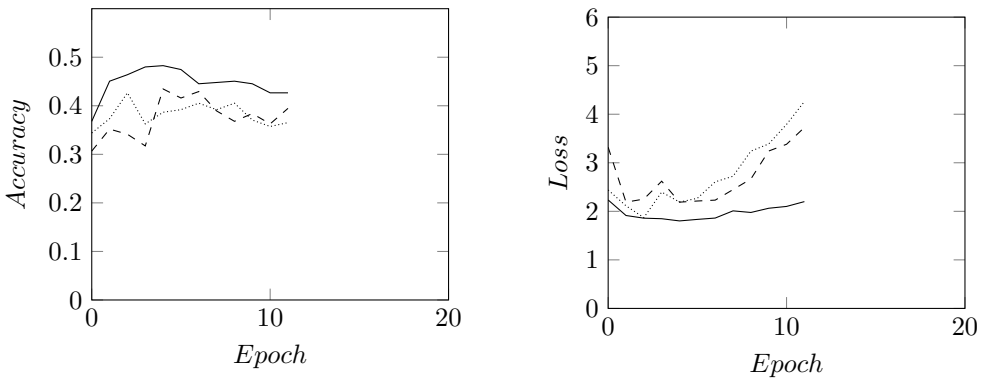
[48] T. Fawcett, "An intoruduction to roc analysis," report, Institute of the Study of Learning and Expertise, 2005.

[49] D. Masters and C. Luschi, "Revisiting small batch training for deep neural networks," report, Graphcore Research, 2018.

Appendices

Appendix A

Optimizer



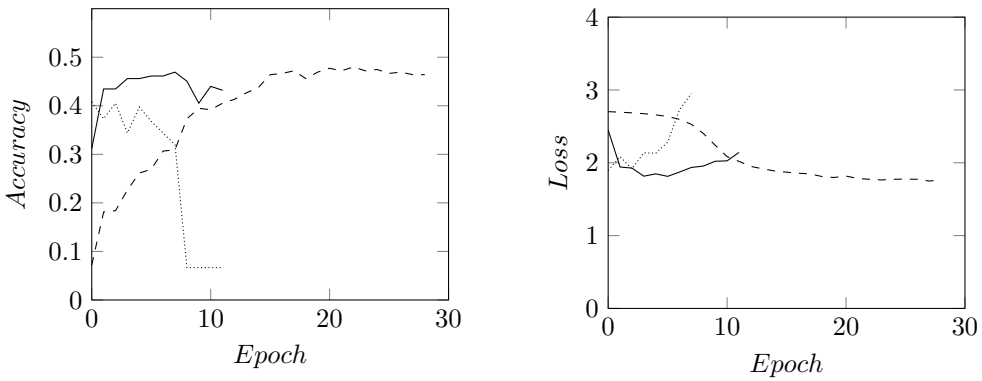
(a) Accuracy

(b) Loss

Figure A.1: Figures of the effect of different optimizers on the CNN model in configuration A: Adam (Dashed), RMSprop (dotted), SGD (Solid); with a learning rate of 0.001 and a batch size of 256.

Appendix B

Learning Rate



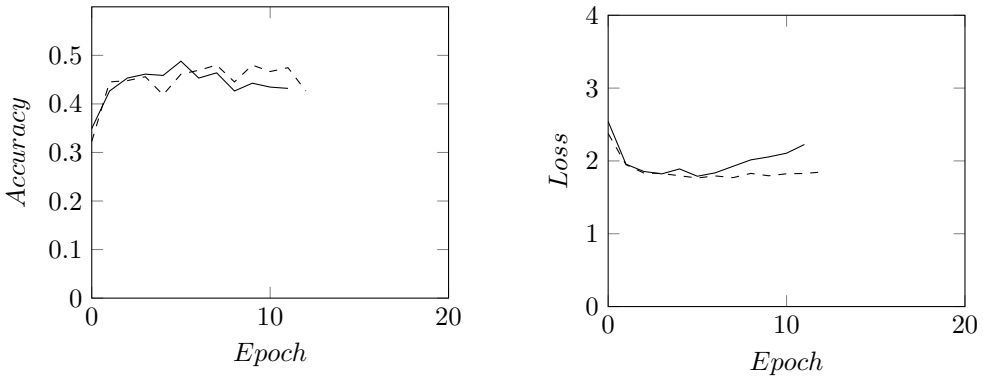
(a) Accuracy

(b) Loss

Figure B.1: Figures of the effect of different learning rate on the CNN model in configuration A: 0.0001 (dashed), 0.001 (solid), 0.01 (dotted); with a batch size of 16 and implementation of the SGD optimizer.

Appendix C

Filter Size



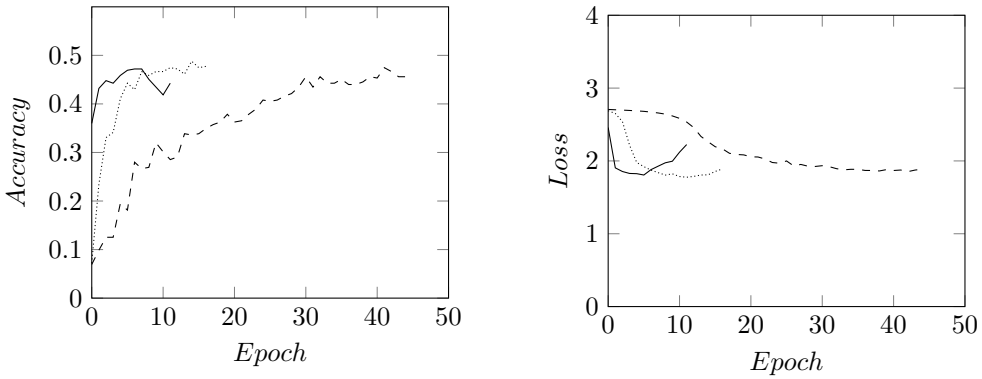
(a) Accuracy

(b) Loss

Figure C.1: Figures of the effect of different filter sizes on the CNN model in configuration A: 3 (solid), 5 (dashed); with a learning rate of 0.001, a batch size of 16 and implementation of the SGD optimizer.

Appendix D

Batch Size



(a) Accuracy

(b) Loss

Figure D.1: Figures of the effect of different batch sizes on the CNN model in configuration A: 256 (dashed), 64 (dotted), 16 (solid); with a learning rate of 0.001 and implementation of the SGD optimizer.