Sound Event Tagging Using Audio and Visual Features

Danny Heard

Supervised by Jon Barker

Module Code: COM3600

This report is submitted in partial fulfilment of the requirement for the degree of BSc Computer Science by Danny Heard

February 16, 2019

Declaration

Abstract

Sound event tagging systems have typically been designed in regard to using audio data as the sole input of the system. However, large new data sources that consist of data from both audio and visual domains such as YouTube are available for audio tagging and sound event detection tasks.

In this project, multi-modal techniques are explored with the aim of enhancing a state-of-the-art audio tagging system by leveraging the visual information from a YouTube based dataset.

These techniques showed that a multi-modal approach to audio tagging is feasible, and improved the performance of sound events that have visual object representations compared to the baseline system. However, further research needs to be carried out in order to address the limitations of the presented techniques.

Acknowledgements

First of all, I would like to thank my supervisor Dr. Jon Barker for supervising this project and for his continuous support and advice.

I would also like to thank my friends and family, for their support and guidance during this year.

Finally, I would like to thank my girlfriend, Marti, for her continuous support and for listening to my mumblings about neural networks.

Contents

1 Introduction **1**

1.1 Aims & Objectives . . . 2

1.2 Summary of Chapters . . . 2

2 Literature Review **3**

2.1 Sound Event Detection . . . 3

2.1.1 System Architecture . . . 3

2.1.2 Evaluation . . . 5

2.2 Feature Extraction . . . 6

2.2.1 Signal Processing . . . 6

2.2.2 Mel-Frequency Representation . . . 7

2.3 Classification . . . 8

2.3.1 Deep Learning . . . 9

2.3.2 Convolutional Neural Networks . . . 10

2.4 Multi-Modal Systems . . . 10

2.4.1 Visual Feature Extraction . . . 11

2.4.2 Integrating Multi-Modal Data . . . 12

3 Audio Tagging Baseline **14**

3.1 Dataset 15
3.2 Feature Extraction 16
3.3 Classification 16
3.4 Evaluation 18
3.5 Setup 19
3.6 Results 19
3.7 Modifying the baseline 21

4 Visual Feature Extraction System 23

4.1 Choice of Visual System 23
4.2 Architecture of Visual System 24
4.3 Video Dataset 25
4.4 Video Dataset Predictions 26
4.5 Integration 27

5 Experiment 1: Early Fusion of Audio and Visual Features 29

5.1 Inception Predictions 29
5.1.1 Setup 29
5.1.2 Results 30
5.2 Reduced Inception Predictions 31
5.2.1 Setup 32
5.2.2 Results 32
5.3 Summary 33

6 Experiment 2: Late Fusion of Audio and Visual Features 35

6.1 Adjusting the Inception Network 35

6.1.1 Setup 36

6.1.2 Results 38

6.2 Decision Fusion 41

6.2.1 Setup 41

6.2.2 Results 41

6.3 Late Fusion of Inception Predictions and Baseline Predictions 42

6.3.1 Setup 43

6.3.2 Results 43

6.4 Summary 45

7 Conclusion **48**

7.1 Findings 48

7.2 Further Work 49

List of Figures

2.1 Mel Filterbank 7

2.2 Generic Neuron 9

3.1 The count for the number of examples for each class in the development dataset 15

3.2 Surrey System Architecture [22] 17

3.3 Baseline system average per class performance showing mean and standard deviation from three runs 20

3.4 Baseline system confusion matrix 20

3.5 Modified baseline average per-class performance 22

4.1 The Inception Module [52] 24

4.2 Inception predictions for the 'train horn' DCASE class 26

4.3 Inception predictions for the 'reversing beeps' DCASE class 27

5.1 Per-class performance for the early fusion all Inception predictions system 31

5.2 Per-class performance for the early fusion reduced Inception predictions system 33

5.3 Confusion matrix for the full predictions early fusion system 34

5.4 Confusion matrix for the reduced predictions early fusion system 34

6.1 The count for the number of examples of each class in the evaluation dataset 39

6.2 Per-class performance of the fine-tuned Inception network 39

6.3 Confusion matrix for the fine-tuned Inception network 40

6.4 Two frames from the dataset with the label ‘Bicycle’ [13] 41

6.5 Effect of weighting on multiplication decision fusion F-score 42

6.6 Comparison of late fusion systems per-class performance with the early fusion and baseline systems 44

6.7 Confusion matrix for the ReLU late fusion system 45

6.8 Confusion matrix for the tanh late fusion system 45

6.9 Examples classified correctly as ‘car passing by’ by the tanh late fusion system [13] 46

6.10 Examples incorrectly not classified as ‘car passing by’ by the tanh late fusion system [13] 46

List of Tables

3.1 Baseline system average overall performance 19
3.2 System performance for the modified baseline 21

4.1 InceptionV3 Architecture 25

5.1 Early fusion performance with the full 1000 prediction vector 30
5.2 System performance using the first 32 principal components of the Inception predictions 32

6.1 Fine-tuning Architecture 37
6.2 Inception fine-tuned performance 38
6.3 Baseline macro and micro-averaged F-score 39
6.4 Individual performance of the two systems to be combined using decision fusion 42
6.5 Decision fusion overall performance 42
6.6 Performance comparison of Inception predictions and baseline predictions late fusion systems 43
6.7 Car passing by class performance system comparison 45

1. Introduction

Machine perception is the field of study that attempts to make machines understand their environment in the same way as humans. There are two main sub-fields: computer vision and machine listening. Computer vision is a well-researched area, with the classic problem being classification and detection of objects or scenes. Visual object recognition systems such as ResNet and Inception-V3 can achieve a top-5 error rate of 3.6% and 3.46% respectively [1], [2], compared to a human top-5 error rate of 5.1% on the ImageNet object detection challenge in which the dataset consists of classes for 1000 different objects [3].

Machine listening includes the problems of speech processing, music and genre recognition, audio monitoring, audio tagging and Sound Event Detection (SED). Audio tagging will be the focus of this project. Audio tagging is where the system detects and produces a list of audio events that occur during a sound clip. SED additionally requires that the onset and offset times for each of the events are detected and listed, therefore audio tagging is a subset of SED. SED has many applications including: security and surveillance [4], searching large video datasets [5], [6], healthcare monitoring [7] and context aware devices such as cars and mobile phones [8]. It is a challenging topic as systems have to deal with variation within the classes and short duration events.

Instead of regarding machine perception and machine listening as independent fields, recently research has been experimenting with combining data from multiple modalities i.e. different sensory domains, to enhance system performance on existing problems. The idea behind this is that most problems in real life span multiple modalities e.g. sound often comes from objects that are moving therefore the object has an effect in both the audio and visual domains. By utilising data from multiple modalities the system can learn better representations of the data [9]. Multi-modal approaches have been used in speech recognition systems to reduce the effect of noise in the audio on the performance of the system [10], [11], which has been shown to perform better than noise reduction techniques [12].

One of the persistent problems in machine learning tasks has been the limited amount of data available to train systems, as it has to be manually collected and annotated which is a time consuming process. However, a solution now exists in the vast amount of data available on the web. The dataset for the present task is audio and video collected from thousands of YouTube videos, it was collated by filtering videos using tags and descriptions to find relevant sound events and then hand labelling the clips with the sound event(s) [13].

1.1 Aims & Objectives

In this project, the aim will be to fuse information from the audio and visual modalities to see if the performance of an existing audio tagging system can be increased and whether it could be a potential topic for future research. Visual features from YouTube videos will be combined with the audio data from the videos.

The project has the following objectives:

- Define an appropriate evaluation method for the aim to be used to evaluate the produced system
- Investigate audio-only event detection and multi-modal techniques and approaches
- Design a system that takes advantage of audio-visual datasets by using a multi-modal approach
- Conduct experiments on the system to evaluate whether an audio-visual multi-modal approach to audio tagging is plausible

1.2 Summary of Chapters

Chapter 2 presents a review of the current research. It starts with an introduction to SED and discusses evaluation metrics, followed by a review of feature extraction and classification used in SED systems, including deep learning and CNNs. It then covers visual recognition systems, finally discussing work in fusing the two modalities. Chapter 3 analyses the baseline audio tagging system used in this project in detail. It discusses the architecture of the system to give an idea of how it will be used in the experiments. The audio dataset is reviewed and then performance of the baseline system is analysed to allow future comparisons to be made and understood. Chapter 4 will review the visual recognition systems discussed in the literature review, one system is chosen to be used as the visual feature extractor in this project. This system is then analysed in detail, before discussing the visual dataset, with some examples run through the visual feature extractor to motivate the use of such a system in this audio tagging task. Finally, techniques of integrating the visual information into the existing audio tagging baseline are discussed, setting out the experimentation that takes place in this project.

The main experimentation is split into two chapters. Chapter 5 explores early fusion techniques, concatenating the visual features with the audio features to then be classified by the audio tagging baseline. Chapter 6 explores late fusion techniques, initially experimenting with fine-tuning the visual feature extractor to output the sound event classes, and then combining the predictions with the audio baseline predictions. Before experimenting with combining the visual features directly with the audio baseline predictions to see which late fusion technique results in better performance. Finally the project is concluded in Chapter 7, summarising the findings and discussing further work.

2. Literature Review

This chapter aims to provide an in-depth discussion of the literature necessary to support the creation of a multi-modal audio-visual audio tagging system. It begins by introducing the problem of audio tagging and sound event detection, exploring the general structure of a sound event detection system and the evaluation criteria for such systems. Afterwards feature extraction and classification techniques for SED systems are reviewed, including a basic introduction to deep learning. The chapter concludes with a discussion of multi-modal systems by reviewing existing visual feature extraction systems that could be used in the project and different approaches to integrating multi-modal information.

2.1 Sound Event Detection

Sound event detection deals with the problem of analysing audio signals to recognise events. It has many applications in surveillance and security, environmental monitoring and context aware devices such as cars and mobile phones [4], [8], [14]. SED involves identifying the onset and offset times as well as the types of events in the audio. Audio tagging is a subset of SED, where only the list of events that occurred in the audio are produced without the onset and offset times for each of the events.

SED can be split into two problems: monophonic sound event detection where only one event can occur at a time, and polyphonic SED where multiple sound events may occur at the same time, overlapping one-another. In real life scenarios, this task is usually polyphonic.

2.1.1 System Architecture

Generally, a SED system is split into a pre-processing stage, a feature extraction stage and a classification stage.

Preprocessing

The preprocessing stage prepares the data for feature extraction. This could consist of formatting the data so it is in the right format, cleaning the data so that missing or incomplete data is fixed or removed.

It could also consist of data augmentation, in which new data is generated from the original data by altering it in specified ways. This is useful when the amount of training data is limited and can also improve the robustness of the system. Examples of data augmentation in SED include pitch transposition and dynamic range compression [15].

Feature Extraction & Normalisation

Features (representations of the data) are extracted from the data that are thought to be useful in differentiating between the different classes of the task. If needed, these features are normalised so they are on the same scale.

Classifier

The classifier uses the provided features to classify the audio into one or more classes. Depending on the way the training data is labelled, the classifier will learn in a different way.

In supervised learning, each example (input) in the training data has a label (output). The classifier then learns a mapping from these inputs to outputs. In unsupervised learning, there are no labels, the classifier infers a function to describe the structure of the data. In semi-supervised learning, which falls between supervised and unsupervised learning, there is a small amount of labelled data compared to a large amount of labelled data.

Weak supervision is a type of semi-supervised learning where one or more labels are present in an example but their specific locations are not known [16]. E.g. for object detection in images, an image is given containing several objects along with a list of the object names. The task would be to identify the location of each object giving the corresponding name.

With the popularisation of deep learning, there has been an ever-increasing need for large datasets since it allows for deeper models and increased performance. One way to satisfy this demand is by using unsupervised or semi-supervised approaches.

2.1.2 Evaluation

Evaluation of sound event detection systems is carried out by comparing the output of a system with a reference for the test dataset, usually called the ground truth. There are different metrics for evaluating the system, which one is chosen depends on which characteristics of the system are the most important.

In polyphonic SED there are two main ways of measuring performance: segment-based metrics and event-based metrics [17]. Segment-based metrics compare the output of the system against the ground truth in short segments. In the DCASE challenge, which is a set of tasks within the field of signal processing aimed to further and promote research within the field [18], one second segments were used. Event-based metrics compare the output of the system against the ground truth event by event. For event-based metrics, a tolerance can be chosen. In the DCASE challenge, a detected event is considered to be correct if the detected onset time is within 200ms of the true onset time and if the offset time is within 200ms or half the length of the event.

Error Rate - Represents the amount of error a system makes.

$$ErrorRate = \frac{Substitutions + Insertions + Deletion}{Number\,of\,Events\,Detected}$$

Where substitutions are detected events with incorrect labels, insertions are detected events that did not occur and deletions are events that did occur but were not detected.

Precision - The ratio of true positives to the total predicted positive.

$$Precision = \frac{TP}{TP + FP}$$

Where True Positives (TP) is the number of times the system correctly predicted the ground-truth label. False Positives (FP) is the number of times the system predicted a label that was not the ground-truth.

Recall - The ratio of positives predicted correctly to the positives predicted incorrectly.

$$Recall = \frac{TP}{TP + FN}$$

Where False Negatives (FN) is the number of times the system did not predict a ground-truth label.

F1 Score - The effectiveness of retrieval (harmonic mean of precision and recall).

$$Fscore = \frac{2 * Precision * Recall}{Precision + Recall}$$

The total scores are calculated by summing each value over all segments, then calculating the corresponding metric.

Generally, optimising for either error rate or F1-score negatively impacts the other. Depending on what the specific application of the system is, one metric might be more useful than the other. F-score is a measure of how much the system detects the reference events. Error rate is a measure of how much the system makes mistakes and is commonly used in speech recognition.

When evaluating the overall performance of a system that deals with more than one class, there is a choice about how to combine the scores from the metrics. One option is to use macro-averaging which calculates the metrics for each class then average to give the overall system performance. The other option is to use micro-averaging which is to add up all the TPs, TNs, FPs, FNs and then calculate the metrics to give the overall system performance. For unbalanced datasets, macro-averaging is used to bias the metric towards the classes with most examples, while micro-averaging is used to bias the metric towards the classes with the least examples [19].

2.2 Feature Extraction

In SED along with most machine listening tasks, feature engineering was the main focus of research and where most performance gains would be made. However, deep learning techniques have demonstrated better performance when using high-level features such as raw waveforms or log-mel spectrograms, than with specifically designed features [20]. Time-Frequency (T-F) representations such as log-mel spectrograms can be passed to a Convolution Neural Network (CNN) and the network will extract the features it thinks will discriminate best between classes .[21].

Most systems submitted to the DCASE 2017 challenge used a T-F representation such as a log-mel spectrogram for the features [22], [23]. One system used a neural network as a feature extractor, passing raw-waveforms through it [24].

2.2.1 Signal Processing

An audio signal must first be sampled in order to be processed by a computer. Sampling converts the audio from its current continuous-time signal form to a discrete-time signal form.

In order to extract features from an audio file, it is split into short periods of time, known as frames, for the duration of which the frequencies are assumed to be stationary. A typical frame size in speech recognition and audio event detection systems is 40ms with an overlap, called the hop-length, of 50%. An audio signal is converted into frames because the frequencies in a signal change over time, and if a whole signal was to be converted from the time domain to the frequency domain using a Discrete Fourier Transform (DFT), it would not capture the change in frequencies over the signal. The technique of splitting an audio signal into frames and computing the Fourier transform on these frames is called the Short Time Fourier Transform (STFT).

A DFT decomposes a signal into the set of frequency components that make up that signal, the number of components to split the signal into is specified, usually 2048. In practice, a Fast Fourier Transform (FFT) is used, which requires the number of components to be a power of 2. A DFT assumes that the signal is periodic and infinite i.e the frame repeats infinitely in the past and future. However, this is clearly not the case for a real world signal as it may be different in the past or future, and is only assumed to be stationary for that period of time. The discontinuities between the edges of the frame can cause frequencies to bleed into other frequencies, creating frequencies that are not actually present. This can be avoided by using a windowing function before applying the DFT to ensure that the values at the edges of the frame are almost zero. The most-commonly used windowing function is the Hamming Window [25].

2.2.2 Mel-Frequency Representation

Mel-Frequency representations are based on how the human ear filters sound frequencies. These representations use the mel-scale, which mimics the fact that the human ear has more filters with a smaller range in the low-frequency region and less filters with a larger range in the high-frequency region. This results in the human ear being able to discriminate better between sounds at low frequencies than sounds at high frequencies.

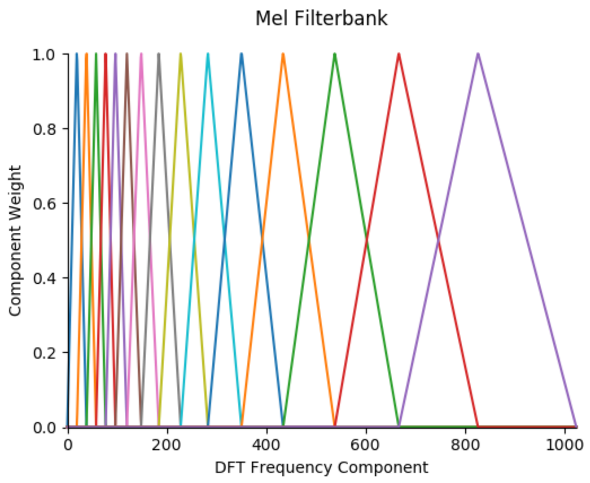

Figure 2.1: Mel Filterbank

The first step is to take the power-spectrum P of the signal which is calculated by using the spectrogram S calculated by the FFT, and applying the formula $P = |S|^2$. The mel filterbank - Figure 2.1, which contains triangular filters, is applied to the power-spectrum to obtain the filterbank energies. The logarithm of these energies are taken to achieve the log-mel spectrogram. The reason why the logarithm is used is because humans do not hear on a linear scale, but a logarithmic scale [25].

A commonly-used feature for speech recognition systems and sound event detections is the Mel Frequency Cepstral Coefficients (MFCCs), which can be computed from the log-mel spectrogram, MFCCs decorrelate the information that is present in the log-mel spectrogram.

2.3 Classification

Prior to neural networks, the main types of classifiers used were Hidden Markov Model (HMM) based classifiers and Support Vector Machines (SVM) [6], [26]. However, due to neural networks ability to learn general representations with less specialised features, and the amount of data now available to train them, most systems now use a type or multiple types of neural network(s) as the classifier [27], [28].

CNNs have become popular in the field as they can learn from the spectrogram of the audio signal, which means that specifically-crafted features do not have to be designed. Another reason they have become popular is that they reduce the impact of noise on the performance of the system [29]. High-performance CNNs, like those used in the ImageNet challenge, can be applied to SED [30]. In the SoundNet system [31], CNNs were used because they are invariant to translations, reducing the number of parameters needing to be learned and furthermore, layers can be stacked allowing high-level features to be extracted from low-level features.

One system submitted to the DCASE 2017 challenge is based on the idea that each event has its own patterns and volume levels in the time scale [24]. It uses an ensemble of CNNs that each take a different length of audio as the input, which helps separate short sounds such as horns from long sounds such as sirens and alarms. The multiple CNNs are used by the system to locate the start and end times of the events for the sound event detection task. In order to capture the temporal information contained in an audio signal, Recurrent Neural Networks (RNNs) can be used [22].

2.3.1 Deep Learning

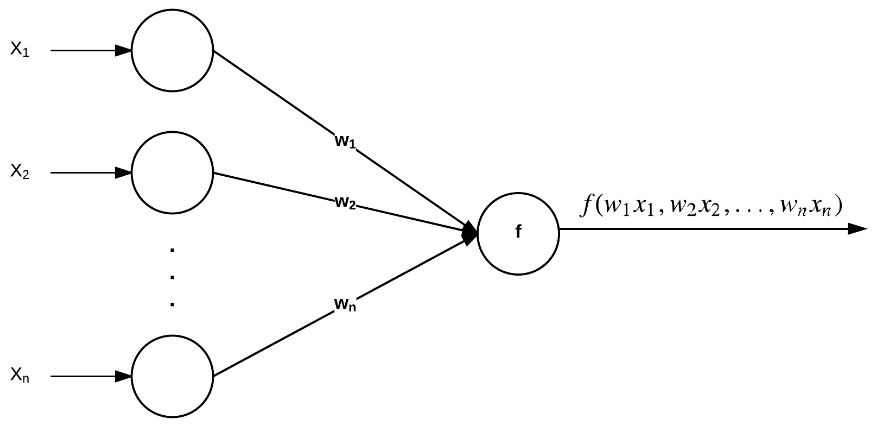

Figure 2.2: Generic Neuron

Artificial neural networks are inspired by the structure of biological neural networks, which are found in animal brains and consist of interconnected neurons. In deep learning, networks are also made up of many neurons. The way the nodes are connected determines how the computation takes place and is an important design decision. Neurons are usually placed in layers, with the outputs of the neurons in each layer becoming the inputs for the next layer. Any layer that is not either the first (input) layer or the last (output) layers are called hidden layers. Each layer learns a more abstract representation of the data than the previous layer.

Figure 2.2 shows the structure of a generic neuron. The neuron takes n inputs $x_1, x_2, .., x_n$, and each input is weighted $w_1, w_2, .., w_n$. The output of the neuron is a function, known as an activation function, of these weighted inputs $f(w_1x_1, w_2x_2, .., w_nx_n)$ [32]. Different activation functions are used in order to achieve certain objectives. Surrey's system used gated linear units (GLUs) as a way to ignore unrelated sounds [22]. Tampere's system used bi-directional gated rectifier units (GRUs) with tanh activation to learn the temporal structure of sound events [23]. The most popular activation functions are sigmoid functions, which have an S curve. The limits for the sigmoid curve are specified by the values (a, b), with common choices being $a = 0$ or $a = -1$ and $b = 1$.

$$\text{Sigmoid activation function:} \quad \lim_{x \to -\infty} f(x) = a \qquad \lim_{x \to \infty} f(x) = b$$

Networks are trained in stages, known as epochs. During each epoch, batches of data are passed through the network, the results are compared against the known 'truth' for the data using a loss function. This loss function along with a gradient-based optimiser, such as stochastic gradient descent (SGD), is used to adjust the weights in the network in a process called back propagation. The rate at which the weights are changed is called the learning rate and is usually small. This is because if the learning rate is too high, it is possible to overshoot the optimal point. However, if the rate is too small then it will take too long to find the optimal point.

2.3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of neural network used for classifying images. CNNs work by initially extracting low-level features and then progressively extract higher-level features using convolutional layers.

Convolutional layers apply a convolutional operation to the input, the input of the layer is the input image if it is the first layer or the output of the previous layer otherwise. A convolution has a specified kernel size and stride value, where the kernel size specifies the area the filter covers, and the stride value specifies how much the filter is moved each time. The moving of the filter across the width and height of the input is called convolving the input. The filter has a weight for each pixel within the area, and the input values in the filter region are multiplied by their associated weights and summed up in the same way as a conventional neural network. Correspondingly, this value is then passed through an activation function to become the value of that neuron. A common activation function in CNNs is the rectifier function, also know as a Rectified Linear Unit (ReLU). A convolutional layer has multiple filters, and when applied creates a number of feature maps, the number of filters is the depth of the layer output [33].

$$\text{Rectifier activation function:} \quad f(a) = max(0, a)$$

Also typical in CNNs are pooling layers which combine the outputs of multiple neurons at one layer into a single neuron at the next layer. They reduce the size of the representation, which in turn reduces the number of parameters and the amount of computation needed. They also help the network gain invariance to rotations and translations of features. The max pooling function is a common pooling function and is where a filter passes across the whole input representation and at each step takes the highest value in a specified region.

2.4 Multi-Modal Systems

The reviewed sound event detection systems uses only audio to classify sound events, however there is motivation to using visual information as well. One example could be sirens, for instance, police cars and ambulances might have similar sounding sirens and therefore, audio-only systems have trouble deciding which class it belongs to. However with visual information it becomes much clearer as to what type of vehicle the siren noise is coming from.

Multi-modal learning is a method that uses more than one modality in order to learn representations [9]. It is an emerging method of learning that is based on the idea that most problems in real life span multiple modalities e.g. sound often comes from objects that are moving, therefore the object has an effect in both the sound and visual domains.

2.4.1 Visual Feature Extraction

Visual features can be extracted from pre-trained visual recognition and detection systems and used to create a multi-modal system. In this section several visual recognition and detection systems are reviewed to investigate how they could be used in a multi-modal context.

Object Recognition

Visual object recognition is the problem of identifying specific objects in images or videos. ImageNet is a huge image dataset containing over 10 million hand-annotated images organised in a category hierarchy structure [34]. It is intended to be used by visual object detection and recognition systems, and has become the de facto standard for training and testing of image object recognition systems.

Since 2010, ImageNet has run a yearly challenge, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [3]. ILSVRC2012 launched the task 'Object Localisation', the task consists of detecting the presence and location of 1000 categories of objects, with the training set consisting of 1.2 million images. Systems submitted to ILSVRC such as ResNet and Inception-V3 can achieve a top-5 error rate of 3.6% and 3.46% respectively [1], [2]. The top-5 error rate e is calculated as follows:

$$e = \frac{1}{n} \sum_k \min_i d(c_i, C_k)$$

Where for each image, $c_i, i = 1, ..5$ is the five class labels predicted the system in decreasing order of confidence, $C_k, k = 1, ..n$ are the ground truth labels for the image with n class labels. Let $d(c_i, C_k) = 0$ if $c_i = C_k$, otherwise 1 [3].

Since 2015, there has been an extra challenge, object detection from video. This is similar to the object detection task, however the dataset only contains 30 categories of objects. Submissions have achieved a mean average precision of 0.7 - 0.8 [35]

Event Recognition

Visual event detection is the problem of identifying events that occur in videos. Unlike object recognition, the uses for such systems tend to be quite specific. There has been work in event recognition for surveillance systems [36], and for human action recognition [37], [38]. These systems are trained on a very small number of event classes, for example, Ji et al's and Ke et al's systems for human action recognition are trained on 3 and 6 classes respectively. These systems categorise an event as some sort of movement between frames.

2.4.2 Integrating Multi-Modal Data

There are two main methods to combine multi-modal data, transfer learning and multi-modal fusion.

Transfer Learning

Transfer learning is when the classifier trains on multiple modalities, but then classifies examples using a single modality. Ngiam showed that when targeting a single modality, transfer learning gives an improvement over using just the one modality [9], transfer learning allows systems to generalise and perform better when training on smaller datasets.

SoundNet is a system based on transfer learning that focuses on the task of recognising scenes and objects from audio [31]. The data consists of unlabelled videos with audio for training and solely audio clips for testing. It uses an ImageNet object detection network, either VGG or AlexNet, to detect objects in the video frames. The output of the ImageNet classifier is used as the target for the output of the sound classification network instead of using the audio labels. The trained models then classify sound data into environments and objects. It was tested on the acoustic scene classification dataset from the DCASE 2016 challenge. It has also been modified and applied to sound event detection [39].

Multi-Modal Fusion

Multi-modal fusion is the process of combining information from multiple modalities in some way, and then using the combined information to make a decision. Information can be extracted from the visual systems by either using the posterior probabilities, or abstract representations from a layer within the network. Speech-recognition systems have used video to improve their performance by tracking mouth movements. It is often used to reduce the effect of noise on the performance of the system [10], [11]. Audio-visual information has also been used in sports event detection and automated surveillance [40], [41].

When using multi-modal fusion, there is a choice about when to integrate the information. Early integration is called feature fusion, and is where the extracted features from each modality are combined into a single representation and fed into the classifier. Late integration is called decision fusion, and is where each modality has its own classifier. The outputs of these classifiers are merged in some way to give the final output [42]. The type of fusion that performs best depends on the data the system deals with. In video analysis, Snoek et. al. showed that different visual concepts performed best with different types of fusion [42]. Lewis and Powers showed that late integration performs better than early integration in audio-visual speech recognition [43]. However Huang and Kingsbury, and Ngiam et. al. showed that early integration outperforms late integration in noisy settings [9], [11].

With early integration, it is not always as simple as combining features from the different modalities. Ngiam found that concatenating audio and visual features can often hurt performance when the audio is not noisy. Instead, he proposed extracting multi-modal features from the raw data as well as audio-only features and then combining them [9].

The Look, Listen and Learn Network (L3-Net) [44] is an example of a multi-modal network that concatenates features from the audio and visual domains i.e. early fusion. However, the audio-visual embedding network [45] that is built on L3-Net, uses an alignment technique to align the two modalities. Hori et. al used an attention based mechanism to combine multi-modal features from the audio and visual domains for automatic visual description [46]. This allowed them to deal with the asynchronous nature of the two modalities. It shows that when working with audio-visual multi-modal data that is not straightforward to fuse them directly at the feature level, some sort of cross-modal alignment is necessary.

3. Audio Tagging Baseline

The DCASE (Detection and Classification of Acoustic Scenes and Events) is a recurrent challenge within the signal processing sub-field of machine learning which aims to promote research in the field, specifically in real-life soundscapes. The DCASE 2017 challenge consists of four tasks: acoustic scene classification, detection of rare sound events, sound event detection in real life audio, and large-scale weakly supervised sound event detection for smart cars [18]. This project focuses on the large-scale weakly supervised sound event detection for smart cars task in which there are two subtasks:

- A. Sound event detection without timestamps (Audio tagging)
- B. Sound event detection with timestamps

The DCASE organisers have provided a baseline system for the task. In subtask A, it had an accuracy of 18.2%, and in subtask B the error rate was 0.93 and a F1 score of 28.4%. The baseline does not train the network in batches, which meant that the system could not run on ShARC, the university's supercomputer, as the resources it required were too large. All submissions performed better than the baseline, therefore a system submitted to the challenge will be chosen as the audio tagging baseline instead.

The top performing DCASE submission was designed by Xu et al at the University of Surrey and has been made open-source [22]. The system has been run on ShARC, in order to run several libraries had to be installed through anaconda (librosa, ffmpeg, libsndfile). In order to extend the existing audio tagging system submitted by the team at the University of Surrey, it will first be analysed in detail.

The structure of this chapter is as follows: it begins with a description of the audio dataset, then the baseline is analysed in detail in the feature extraction and classification sections. The evaluation method is described before the baseline system's set-up and performance is discussed. The chapter ends with a explanation of how the baseline has been modified to suit this project.

3.1 Dataset

The task uses a subset of the AudioSet dataset [13], which is a collection of over 2 million labelled 10 second clips drawn from YouTube videos covering 632 sound events. The clips are collated from thousands of YouTube videos, using tags and descriptions to filter videos with relevant sound events, the clips are then hand labelled with the sound event(s). This subset contains 17 sound events divided into warning sounds and vehicle sounds. The data is split into two sets, training which has weak labels, and testing which has strong labels. There is also a separate evaluation dataset with strong labels. There are 140 hours of audio in the development dataset. The training set contains 51,172 audio clips, the test test contains 488 clips and the evaluation set contains 1,103 clips. It should be noted that 10 seconds is the limit for the length of the clip, there are 10,785 clips that are shorter than 10 seconds in the training set.

- Warning sounds: Train horn, Truck horn, Car alarm, Reversing beeps, Ambulance siren, Police siren, Fire truck siren, Civil defence siren, Screaming
- Vehicle sounds: Bicycle, Skateboard, Car, Car passing, Bus, Truck, Motorcycle, Train

The classes are not balanced within the training dataset - Figure 3.1, there are many more 'car' examples than any other class, which means that any system using this dataset may overfit on the 'car' class. In the test dataset, each sound event appears in at least 30 clips and in the evaluation dataset, each sound event appears in at least 60 clips.

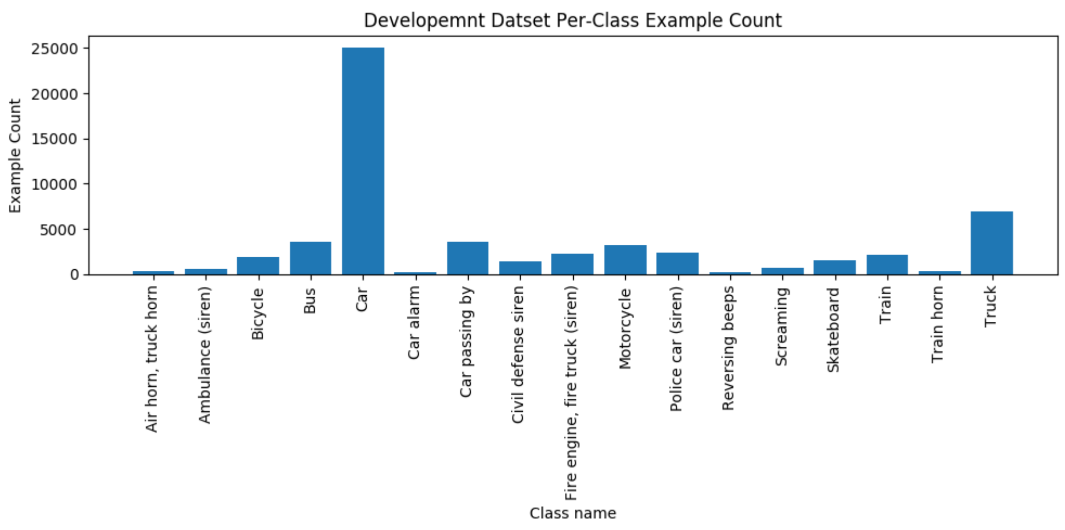

Figure 3.1: The count for the number of examples for each class in the development dataset

3.2 Feature Extraction

The feature extraction method converts the audio file into a 240 X 64 feature matrix, which consists of 240 log-mel spectrograms each with energies at 64 filter positions. The 240 log-mel spectrograms span the whole of the 10 second audio file, this allows the temporal change to be captured by the network itself, rather than becoming an extra feature.

The audio is initially preprocessed, which consists of downsampling the audio to 16kHz and converting the audio from a multi-channel to a single channel recording by taking the average of the channels and then treating it as a single channel recording.

The system uses the spectrogram function provided by the scipy package to compute the necessary transformations, the parameters of which have been set by heuristic knowledge [47]. The length of each segment is 64ms (1024/16000) with a 35% overlap. The number of points (frequency components) for the FFT is 1024, and a hamming window is used for the windowing function. The spectrogram function is set with the 'magnitude' option, so that it returns the power spectrum of the signal. To compute the log-mel spectrogram, a 64 length mel-filter bank is used with a frequency range of 0 to 8kHz, which is half the sampling rate, due to the Nyquist theorem. The dot product of the power spectrum with the mel-filter bank is taken, and the the logarithm of this is taken to give the final log-mel spectrogram.

The features are then normalised by computing a scalar from the training data using the scikit-learn StandardScalar function, this function creates a scalar by removing the mean and scaling to unit variance. This scalar is then applied to the training and testing dataset to normalise the data.

3.3 Classification

Figure 3.2 shows the structure of the network. The convolutional layers are applied to the log-mel spectrogram to extract high level features, then a bi-directional recurrent neural network (Bi-RNN) is used to capture the temporal information. Finally, a feed-forward neural network (FNN) is applied to predict the posteriors of each audio class at each frame. The prediction probability of each audio class is obtained by averaging the posteriors of all the frames.

The convolutional layers use gated linear units (GLUs) as the activation function. The GLUs are used in this system to introduce the attention mechanism to the neural network, that is, the network will learn to attend to the audio events and ignore the unrelated sounds [22]. The GLU is defined as:

$$\mathbf{Y} = (\mathbf{W} * \mathbf{X} + \mathbf{b}) \odot \sigma(\mathbf{V} * \mathbf{X} + \mathbf{c})$$

Where σ is the sigmoid non-linearity, $*$ is the convolution operator and $\odot$ is the element wise product, $\mathbf{W}$ and $\mathbf{V}$ are the convolutional filters, $\mathbf{X}$ is the input and $\mathbf{b}$ and $\mathbf{c}$ are

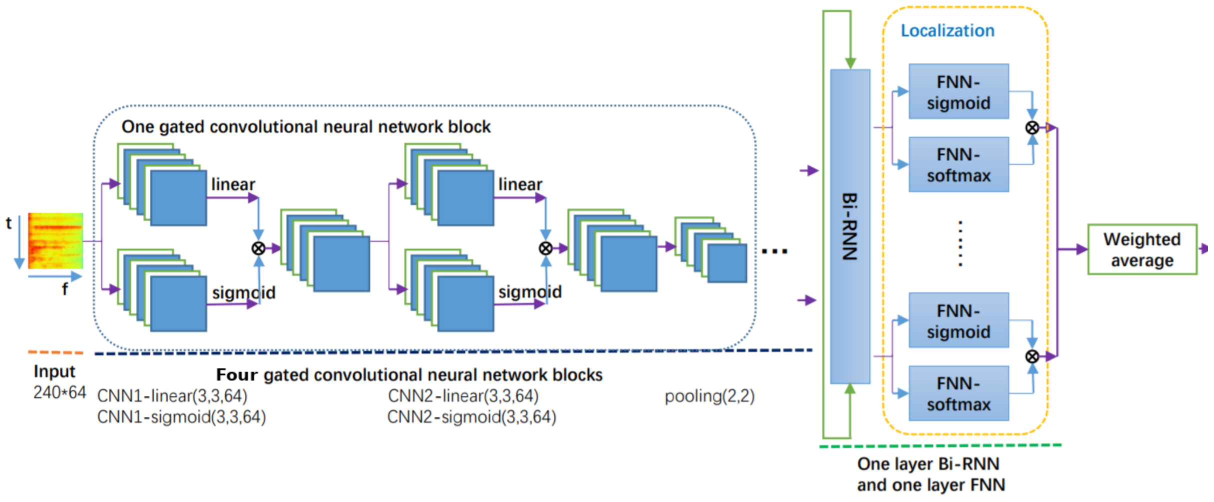

Figure 3.2: Surrey System Architecture [22]

the biases. This can be implemented by taking the element wise product of a convolutional layer using a linear activation function with a convolutional layer using a sigmoid activation function.

There are four convolutional blocks, with each block using the output from the block before, and the first block using the input to the network, the log-mel spectrogram. Each block consists of two convolutional layers with gated linear units followed by a max pooling layer to reduce the dimensions of the representation and to extract robust features against the background noise [47]. After the four blocks, a final convolutional layer using the rectified linear activation function is applied followed by a max pooling layer.

The features extracted by the CNN are fed into the Bi-RNN to select related information for each audio event from the long-term context, the bi-directional nature of the layer means that future information as well as past information can be used [47]. The Bi-RNN consists of a single bi-directional recurrent layer using a GLU as the activation function. The following FNN maps the representation onto the output classes. The FNN is also used for the system to localise the audio events for SED, but as SED is not the focus of the topic, the method will not be discussed.

The loss function used for training the network is binary cross entropy because it was shown to perform better than the mean squared error for labels with zero or one values [47]. Binary cross entropy is defined as:

$$E = -\sum_{n=1}^{N} (\mathbf{P}_n log \mathbf{O}_n + (1 - \mathbf{P}_n) log(1 - \mathbf{O}_n))$$

where E is the binary cross-entropy, $\mathbf{O}_n$ and $\mathbf{P}_n$ denote the predicted and reference label vector for the sample n respectively, N denotes the batch size. The network uses the Adam optimizer for back propagation [48]. After each epoch, the model weights are saved.

The system employs a batch data balancing method to counter-act the highly unbalanced training dataset, as it could be possible that a batch consists of examples from a single class. It does this by ensuring that the frequency of different classes in each batch are balanced, so that the number of most frequent samples is at most 5 times than the number of least frequent samples in a batch.

To predict the events for a clip or a set of clips, the system computes the class probabilities from each of the last ten epochs of the trained model, then the average of these probabilities is taken. A threshold value is set at 0.3 for all the sound events, if the predicted probability of the audio clip containing the event is higher than this, then the audio clip is classified as containing that event. A threshold must be set because of polyphony as multiple classes can apply to a single audio clip, therefore all classes in the problem will have some probability of being in a specified audio file.

3.4 Evaluation

This project is focusing on subtask A - audio tagging, this is because for subtask B - SED, how the system deals with the weakly labelled data is a concern. For subtask A this is not an issue as the precise location of the events does not need to be found, it only needs to be stated if an event is present in the audio. This reduces the complexity of the task and therefore makes it possible in the course of this project.

This project will be using the same evaluation method as the DCASE challenge, this method is explained in subsection 2.1.2. As the focus is on subtask A, the evaluation metric used is F-Score. By using the same evaluation metric as the DCASE challenge, the produced system can be compared against the DCASE submissions to evaluate whether there was any improvement by using the video data.

$$Fscore = \frac{2 * Precision * Recall}{Precision + Recall}$$

For each audio file in the dataset, the system will output a list of predicted audio events for that audio clip. The ground truth contains a list of the files each with the actual audio events that occurred within that clip. True Positives (TPs) are calculated by counting all the events that the system predicted occurred in the audio clips that did occur in the audio files as stated by the ground truth. False Negatives (FNs) are calculated by counting the number of events that occurred as stated by the ground truth that were not predicted by the system. False Positives (FPs) are calculated by counting the number of events the system predicted for the audio clips that were not stated in the ground truth. Precision and recall can be calculated using these values and then the F-score can be calculated.

3.5 Setup

The baseline system was set-up by precisely following the instructions set in the README.md given by the researchers, and the audio dataset was downloaded using the download_audio.py file provided by the organisers. The system will be run three times in order to see the variance of the system as this information will be useful when comparing results of experiments as increased performance could just be natural variation within the system and not a significant improvement.

The batch size is set to 44 and the number of steps per epoch set at 100, that is after 44 training examples are seen, the weights will be updated. The researchers have found 31 epochs to be the optimal amount by manual selection.

To evaluate the system on the evaluation dataset and obtain the performance metrics for the audio tagging subtask, the evaluation code provided by the organisers was downloaded [49]. It was also modified to print out the results in a cleaner format so that it is easier to use the results for graphs and tables.

3.6 Results

Table 3.1: Baseline system average overall performance

	Precision	**Recall**	**F1 Score**
First Run	54.6	57.0	55.8
Second Run	55.5	58.1	56.7
Third Run	57.0	59.2	58.1
Mean	$55.7 \pm 2.2\%$	$58.1 \pm 1.9\%$	$56.9 \pm 2.0\%$

Table 3.1 shows the overall mean performance and variance of the baseline system over three runs. The mean performance of the baseline will be used to compare against further experiments as well as taking into account the variance of the system in order to rule out insignificant improvements. There is an amount of variance in the performance because in a neural network, connection weights are initialised randomly before training unless specified. Since the final connection weights in a network will differ slightly depending on the initial weights, as they will converge to different local minima, and the initialisation of the weights is random, there will be an amount of variance in the results when run multiple times.

As Figure 3.3 shows, the system has poor variance on the 'ambulance siren' class. The baseline has poor performance on the 'car passing by' class with an F-Score of around 20%, yet it obtains a good F-score of around 60% on the 'car' class. It is plausible that the system is mistaking one class for the other as they have very similar sounds but only the change of the sound over time makes a distinction between the two. Therefore, by

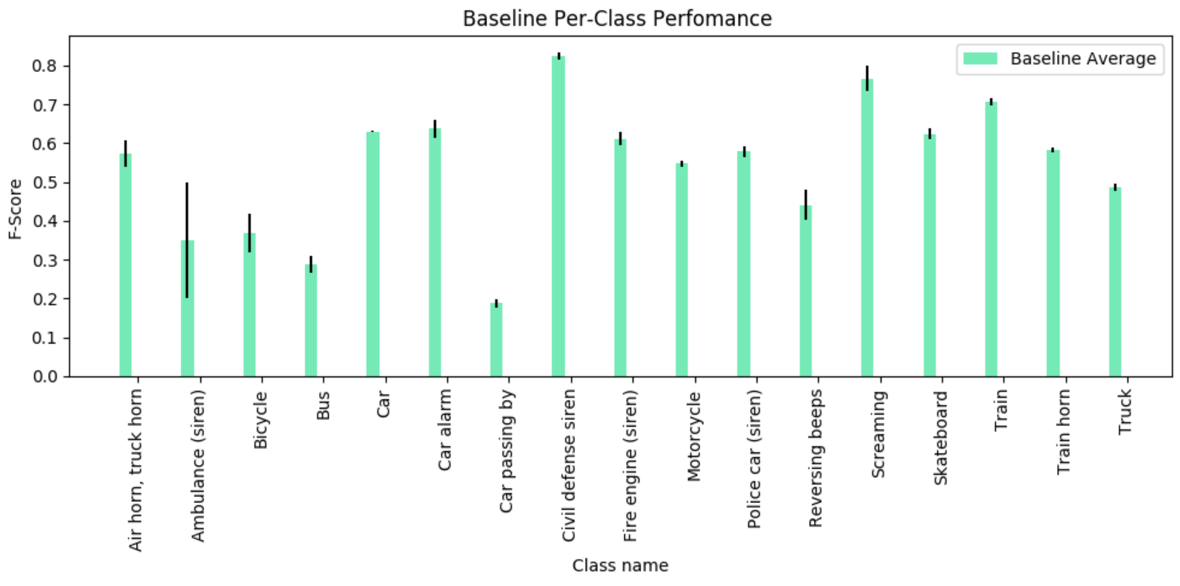

Figure 3.3: Baseline system average per class performance showing mean and standard deviation from three runs

including visual motion information e.g. by taking visual features over multiple frames, it might be possible to decrease the confusion between the two classes. 'Bus', 'bicycle' and 'reversing beeps' are other classes with a lower F-Score than the average.

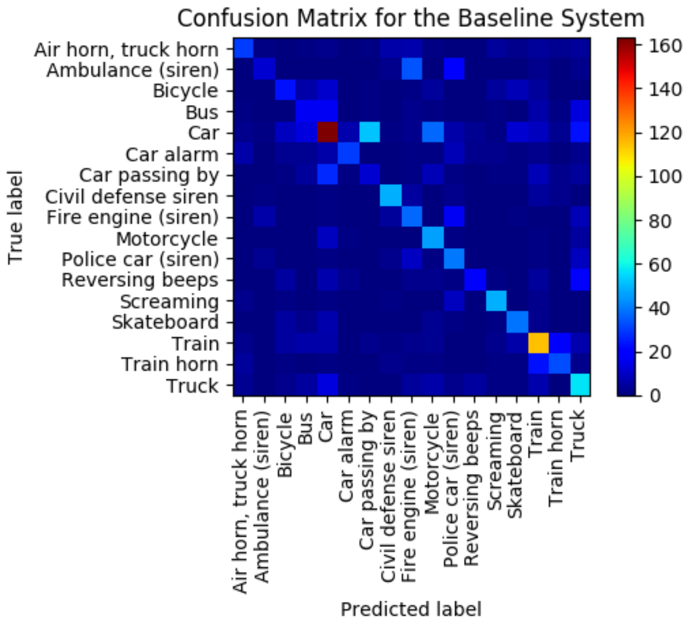

Figure 3.4: Baseline system confusion matrix

If we look at the confusion matrix in Figure 3.4, it shows that the system is confusing the 'car passing by' with the 'car' class, which explains the poor performance on the 'car passing by' class. Also interesting is that the system is confusing 'truck' with many other classes such as 'car' and 'bus', therefore with the added visual information it is expected that the system will not confuse these classes as much as it currently does. Additionally, it can be seen that the emergency service siren classes are being confused with each other, which again, might be expected to be improved when using the visual information.

3.7 Modifying the baseline

In order to be up-to-date and able to work with the latest packages and versions, the Surrey system, originally written in Python 2 was converted to Python 3 using the 2to3 tool. Parts of the code that read and write to/from data files had to be updated manually as the way Python handles files changed from version 2 to 3.

In order to expand the existing system to be able to deal with more data, the way it loaded data had to be made more efficient, as out of memory issues cropped up when using a larger number of features, as is the case in the early fusion chapter. To do this, the examples are read sequentially from the hdf5 file in which they are stored, instead of loading all the examples at once. Also, in the baseline system the features are scaled in-memory before being passed to the neural network. Since the whole dataset cannot be loaded into memory in the expanded system, the features are scaled once during feature extraction, and in the same sequential manner as discussed.

To show that this updated system did not cause any negative effects on the performance of the system, it was ran and the results are compared to the original baseline system.

Results

Table 3.2: System performance for the modified baseline

Precision	**Recall**	**F1 Score**
56.65	57.4	57.6

The overall precision, recall and F1 score are all within the range for the original baseline, therefore modifying the baseline has not affected the performance of the system.

Figure 3.5 shows that most of the class performances are within the standard deviation of the original baseline. While some of class performances, such as ‘police car siren’ and ‘train horn’ are outside of the standard deviation of the baseline system, they are still within the range of the original baseline.

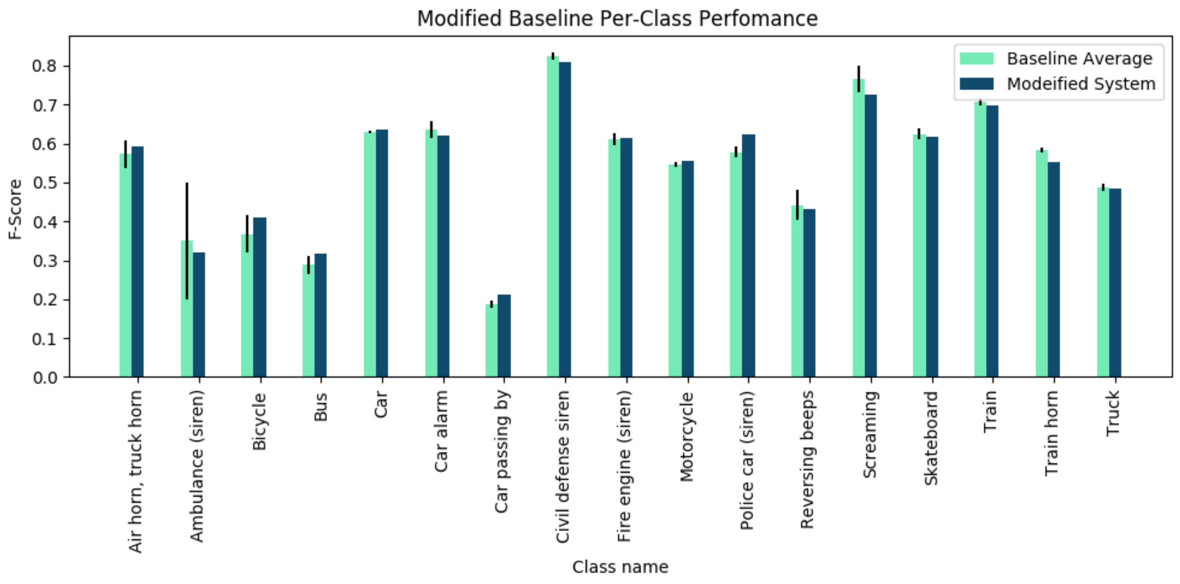

Figure 3.5: Modified baseline average per-class performance

4. Visual Feature Extraction System

The system will use an existing visual classification / detection system in order to extract visual features to be used alongside the audio features.

The structure of this chapter is as follows: it begins by discussing existing visual classification and detection systems in regard to their suitability in extracting visual features for use in this project. Then the architecture of the chosen system is described, before examining the video dataset and exploring the performance of the chosen system on the video dataset. Lastly, methods of integrating the extracted visual features into the baseline are considered leading to describing the experiments that will be carried out during this project.

4.1 Choice of Visual System

Here the two types of system that were discussed in the literature review will be compared in regard to their feasibility in this project.

Event recognition applies more intuitively to this problem than object recognition, for the reason that if the system can tell the difference between a moving vehicle and a stationary vehicle, the sound is more likely to come from the moving vehicle. As an example, if you look at an image of a scene and then at a silent video of the same scene, it is easier to imagine the sound from the video than from the image. However the current research is video event detection is minimal, limited to small-scale applications and there are few open-source systems. The systems researched in subsubsection 2.4.1 only detect events that consist of some sort of movement, however there are sound-event classes in the DCASE challenge that do not correspond with movement in the video e.g. car alarm. In these cases the video event detection system will return a false negative.

Systems from the ILSVRC image object detection task are trained on 1000 classes of objects, and therefore have the largest chance of being applicable to this task rather than systems from the ILSVRC video object detection task, which are only trained on 30 classes of objects. The implementation of some ILSVRC image object detection systems are available along with pre-trained models in Keras [50]. Keras also includes functions that allow the models to be trained from scratch and furthermore, to fine-tune the systems to fit the desired application. Therefore this project will use an image object detection system.

Keras has multiple ImageNet architectures available with pre-trained weights. These are Xception, VGG16 and VGG19, ResNet50, Inception-V3 and Inception ResNetV2, MobileNet, DenseNet and NASNet. The Inception network offers good performance on the ImageNet dataset, while at lower cost than many other networks, i.e. the number of parameters is fewer so it is quicker to train and classify images [51]. The size of the network is 92mb with 23,851,784 parameters, compared to VGG16 which is 528 MB and has 138,357,544 parameters. Inception is chosen for this reason, because when fine-tuning the network the lower computation will result in quicker training time which is necessary in this project as it must be completed in a short time frame. Inception has been run on ShARC to show that all requirements are present.

4.2 Architecture of Visual System

The architecture of the Inception network will not be analysed in detail as it is out of scope for this project, however further details can be found in the Inception paper [2]. Instead a small overview will be given, stating the most important points for this project.

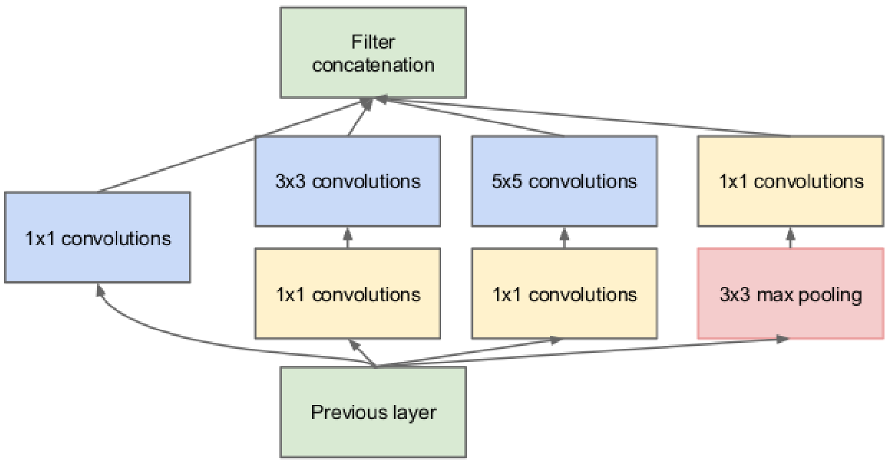

Figure 4.1: The Inception Module [52]

The main concept of the Inception network is the Inception module - Figure 4.1, which applies multiple convolutions of different sizes to the same input before then concatenating them [52]. This allows the model to learn which convolutions apply best to different objects. The module also performs 1x1 convolutions before the larger convolutions to reduce the dimensionality of the input, so the number of operations and therefore computation time is much smaller than it would otherwise be.

Table 4.1: InceptionV3 Architecture

Type	**Patch size/stride**	**Input size**
conv	3x3/2	299x299x3
conv	3x3/1	149x149x32
conv padded	3x3/1	147x147x32
pool	3x3/2	147x147x64
conv	3x3/1	73x73x64
conv	3x3/2	71x71x80
conv	3x3/1	35x35x192
3x Inception		35x35x288
5x Inception		17x17x768
2x Inception		8x8x1280
pool	8x8	8x8x2048
linear	logits	1x1x2048
softmax	classifier	1x1x1000

Table 4.1 shows the architecture of the InceptionV3 network. There are three groups of Inception modules each with different convolutional methods which are more efficient for their specific grid size. The network concludes with a softmax layer that maps the representation onto the 1000 ImageNet classes, with a prediction probability given for each one of the classes.

4.3 Video Dataset

The video dataset comprises of the corresponding YouTube videos from which the audio clips are taken. To download the video dataset the download_audio.py provided by the organisers of the challenge was modified to download the video clips without audio. As the process had to download the entire video from YouTube and not just the necessary 10 second clip, due to the way YouTube videos are downloaded using the pafy Python library, it would take a very long time to download the full 51,172 video dataset sequentially. Therefore the script was converted to take advantage of multi-processing so that the download process could take place in parallel and speed up the time it would take to download the full video dataset. However it was not possible to download all the 51,172 videos due to some videos being deleted or taken down since the initial dataset was released, consequently the number of videos in the training, testing and evaluation sets is 41,539, 398 and 984 respectively. Further, the evaluation dataset did not include a list of the videos like the training and testing datasets had, therefore this had to extracted from the evaluation ground truth list in order to download the evaluation videos.

In the dataset there are some videos that do not correlate with its sound event class, such as lights and a speaker on a printed circuit board and background noise in a video where the sound-emitting object does not appear. It is hoped that the system will learn to ignore these videos, but it is possible that some classes might perform worse when using the video data, as some of the videos for that class are unrelated to the sound.

Since the Inception network is an image object detection system and the visual dataset is comprised of videos, frames must be extracted from the videos to be classified by the Inception network. There are several ways of doing this, the simplest being taking one frame from a specified point in all of the videos using ffmpeg. Another more intriguing option is to take a frame from the video each time the scene changes, which could be possible using the PySceneDetect library [53].

4.4 Video Dataset Predictions

In order to motivate using Inception as a visual feature extractor for the audio tagging task, the middle frame from the videos of some of the DCASE classes will be classified by the network, the top three predictions will be taken and plotted on a bar chart. This will allow comparisons between the objects in the video frames classified by the Inception network and the DCASE labels to show that relevant objects are being observed in the frames. It is not possible to show all of the classes, as the high number of videos for some of the classes resulted in a large number of predicted classes, and therefore the graph is too large to display in this report. However, some of the classes with a lesser number of examples are shown instead to motivate using the Inception network.

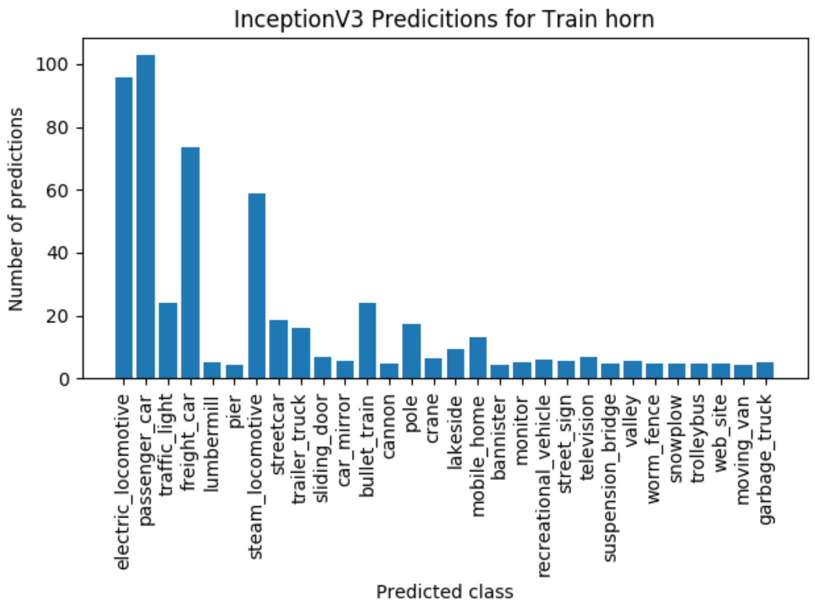

Figure 4.2: Inception predictions for the ‘train horn’ DCASE class

As Figure 4.2 and Figure 4.3 show, for the DCASE audio classes, the Inception network generally recognises objects that are similar or related to the DCASE class. For example, with the ‘train horn’ DCASE class - Figure 4.2, the highest number of predictions were for the ‘electric locomotive’, ‘passenger car’, ‘freight car, and ‘steam locomotive’ ILSVRC classes, all of which are objects that are related to a train. The audio baseline performed

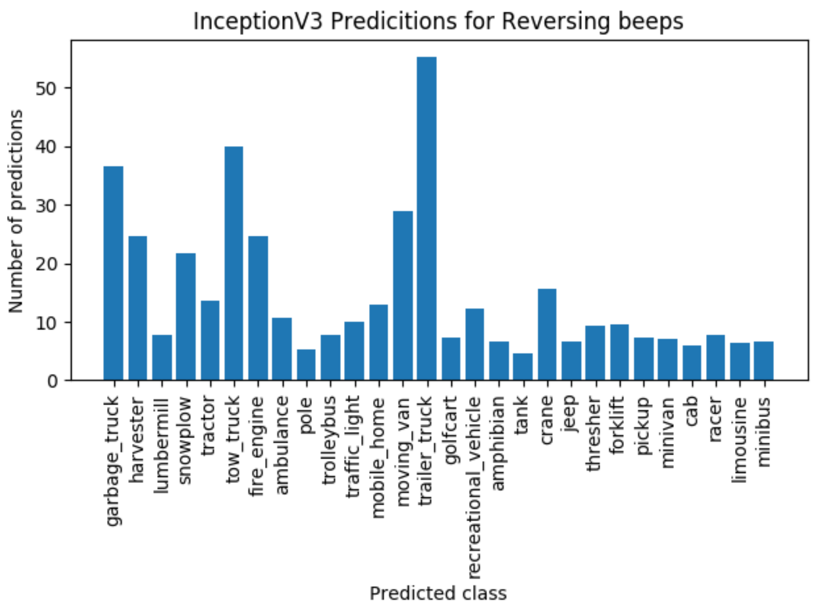

Figure 4.3: Inception predictions for the 'reversing beeps' DCASE class

poorly on the DCASE class 'reversing beeps'. However if we look at the Inception predictions for this class - Figure 4.3, again the highest predictions are for objects related to the audio class e.g. 'trailer truck', 'tow truck' and 'garbage truck'. Therefore by employing this information, the hope is that the performance on these poor performing classes can be increased.

4.5 Integration

There are several options for the integration of the two systems, which are generally split into early and late fusion. Previous work in multi-modal fusion - see subsubsection 2.4.2, found that different visual concepts respond better to different types of fusion [42]. By experimenting with different types of fusion in this project, we can see how this data responds to different types of fusion.

The simplest way to use the predictions from the Inception network is to use them as features and combine them with the audio features to be classified by the audio baseline. This is called feature fusion, a form of early fusion. This data may need to go through a dimensionality reduction process such as PCA, as there are 1000 ImageNet classes which might be too much data for the audio system to handle. The Inception system could also be trained from scratch on frames from the DCASE video data before doing feature fusion, so it fits exactly to the dataset. However the amount of data available is much smaller than the ImageNet training data and therefore would not necessarily result in better performance on the DCASE video data. Furthermore, because the Inception network is very deep it is possible it might overfit on the training data. It would also take time and resources which would be better spent elsewhere.

Another option is to remove the output layer of the Inception system and build a new classifier on top of the network where the output is the 17 DCASE classes. Then train that classifier using the video frames, in a method called transfer learning. Transfer learning can be taken one step further in a process called fine-tuning, where the weights of the final layers of the Inception network are also updated when the new classifier is trained. The choice between fine-tuning and transfer learning generally depends on the size of the new dataset, and the similarity to the ImageNet dataset. Once the output of Inception network is the 17 DCASE classes, either by fine-tuning or transfer-learning, then decision fusion - a form of late fusion, can be used to integrate the visual information. One way of performing late fusion is to combine the probabilities by averaging them or multiplying them together. Another way is to build a third network, with the input being the outputs from both systems, train this final network so it will output a final decision.

Each subsystem will be separate, with results being saved to a file and read by the next subsystem. This allows changes to be made to one subsystem without affecting the other subsystems, therefore the lengthy retraining of all the systems is not required. This also gives a benefit in that the systems will not need to be linked together in one code-base, meaning if certain systems are implemented in a different programming language or using different libraries, they do not have to be converted to the same language and framework, which could potentially create issues or bugs.

5. Experiment 1: Early Fusion of Audio and Visual Features

Early fusion, also called feature fusion, is the combination of features from multiple-modalities before passing them to the network. The idea is that the network will learn high-level representations that cross-over in both domains, so that the representation is more accurate than it would be if it was trained on a single modality.

5.1 Inception Predictions

As a first experiment at creating an early fusion multi-modal system, the full output vector of the Inception network will be concatenated with the audio features and then the system will be trained on this large feature set. The input frame to the Inception network will be a frame from the middle of the video, so the predictions will be for the objects in the middle frame of the video. The middle frame was chosen because it allows the camera to steady after the start of the video, therefore the detection of objects may be more accurate.

5.1.1 Setup

The output of the Inception network is a 1 X 1000 vector, representing the posterior probabilities of every ImageNet class for the frame. Since the input to the audio system is a 240 X 64 matrix, representing 64 frequency bins computed from the sampled 64ms frames including a 35% overlap, giving a 240 length time dimension. In order for the Inception posterior probability vector to be concatenated with the audio features it must be the same size as the audio features matrix. Therefore the predictions are repeated 240 times to give a 240 X 1000 matrix, which is then concatenated along the horizontal axis. By repeating the vector along the time axis, we have assumed that the predictions will not change in time, however this is not the case and is subject to a separate study.

There are a few problems with this naive implementation of an early fusion multi-modal system. Firstly, the visual information matrix is 16x bigger than the audio matrix and therefore it may misguide the network into thinking the video features give more information about the audio events than the actual audio information. Secondly, because the baseline's network uses a convolutional layers with a kernel size of 3x3, it captures features that are derived from the change in the mel-frequency spectrograms over time. Consequently, as the frame predictions are stationary over time, the network will not be able to find time features. So the network will be trying to learn parameters that represent change over time for the audio data, but also parameters that represent the constant data of the predictions. This mismatch might mean the network cannot learn good parameters and therefore reduce the performance compared to the baseline system.

To handle the extra data, the network had to be slightly modified as the representation size in the network had to be decreased. This is because the output of the convolutional blocks was too large for the input to the Bi-RNN. This can be done by increasing the size of the pooling layer in each of the convolutional blocks, so that the representation size is reduced in parts. Since there are four convolutional blocks, the pooling layer of each block was increased to (1,4) from (1,2), which is a 2x increase over four blocks i.e. $2^4 = 16$, which increases the size of the network by 16x. The new input vector length is 1064, 64 audio bins + 1000 Inception Predictions, 1064 / 64 = 16x increase. The batch size also had to be decreased, as 44 examples of 1064 length was too large for the GPU to store in memory. The batch size is decreased to the largest power of 2 that the GPU can handle, which is 8. The steps per epoch value was increased in proportion to the decrease in batch size.

5.1.2 Results

Table 5.1: Early fusion performance with the full 1000 prediction vector

Precision	**Recall**	**F1 Score**
53.7	58.5	56.0

Comparing the performance of the full predictions early fusion system - Table 5.1, to the baseline system - Table 3.1. The precision of the early fusion system is lower, but recall and F1 score are within the range of the baseline system. The decrease in precision can be attributed to the fact that the Inception predictions might contain predictions for objects in the frame that are not the related to the sound event. For example, in a busy frame the only sound event might be 'bus', yet the frame could contain cars, bikes and other objects.

By analysing the per-class performance - Figure 5.1, it is interesting to see the early fusion system vastly outperformed the audio baseline on the 'bicycle' and 'bus' classes. Unfortunately, the improvements on these classes were not enough to offset the drop in performance on the other classes, specifically the 'car alarm', 'reversing beeps' and 'screaming' classes. The reason the early fusion system may be performing better on these classes is that they are sound classes that match a visual object, so the Inception network is able to 'see' those classes and pass it to the audio system as a feature. Furthermore the classes the early fusion system underperforms against the baseline system e.g. 'reversing

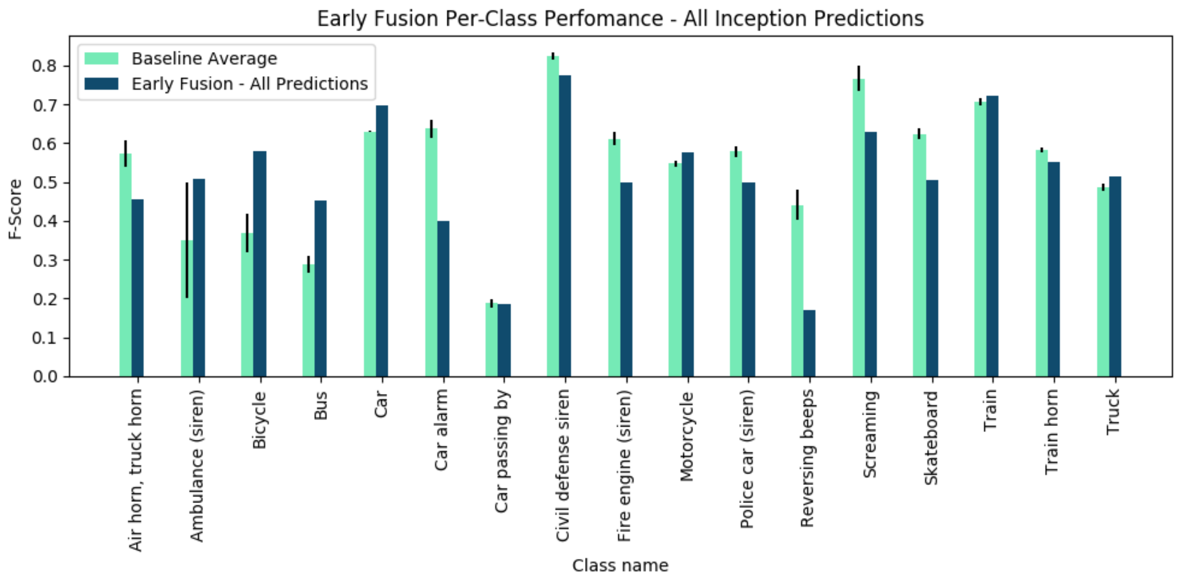

Figure 5.1: Per-class performance for the early fusion all Inception predictions system

beeps' and 'car alarm' are sound-specific classes that do not map into a specific visual object. Therefore, the system is not learning to ignore the information from the visual features for the classes that do not have a representation in the visual domain.

If we compare the confusion matrix of the full predictions early fusion system - Figure 5.3 with the baseline - Figure 3.4, the early fusion system is not predicting as many sound-only labels, such as 'car alarm' and 'reversing beeps'. It shows that by integrating visual data using the full Inception predictions, the system becomes less confident about classes that only have a representation in the sound domain and not in the visual domain.

5.2 Reduced Inception Predictions

One of the problems with the initial early fusion technique of using the Inception predictions is that the prediction vector is 16x bigger than the audio feature vector. The length of the feature vector should represent the amount of data that feature gives to the problem. Since this is an audio event detection problem, video data should give less information to the system than the audio features, therefore the number of dimensions should be in the region of $20 \sim 30$. Since the length of the audio feature vector is 64, the chosen length of the visual vector is 32, which allows the size of the network to be increased easily as it is an increase of a third.

To reduce the size of the visual information, the prediction vector from the Inception system will undergo Principal Component Analysis (PCA) to reduce the data to the dimensions that give the most information about the data. PCA transforms the data into a set of orthogonal dimensions (components) in which the first dimension has the largest possible variance, and each following dimension has the largest possible variance while being orthogonal to all previous dimensions. To conduct principal component analysis on a set of data, it is first normalised by subtracting the mean from each of the data points, so that the mean is zero. The covariance matrix is then calculated, from which the eigenvectors and values are computed. The eigenvectors are sorted by their eigenvalue, giving the sorted principal components of the dataset. A number of components are kept, and the dot product of these with the normalised data is taken to give the reduced data.

5.2.1 Setup

Again because the data is larger than that of the original baseline, the size of the network had to be increased. Since the input is only increasing by a third, it should be necessary to increase just one of the pooling layers in the network rather than all of them. Therefore, which layer to increase will be the focus of some experimentation. Since the data is increasing by a third, and the pooling layer size for each convolutional block is (1,2), it has to be increased to (1,3). Or for the final layer, the one before the Bi-RNN, which has a size of (1,4) needs to be increased to (1,6).

The batch size could be increased in this experiment as the input is smaller and therefore more data could be handled by the GPU. But by keeping the batch size and steps per epoch the same as the previous experiment, it will be possible to compare the two different methods rather than the architecture of the model.

5.2.2 Results

Table 5.2: System performance using the first 32 principal components of the Inception predictions

Pooling Layer	**Precision**	**Recall**	**F1 Score**
First block pooling layer	41.7	45.2	43.4
Second block pooling layer	42.2	45.1	43.6
Third block pooling layer	38.5	44.0	41.1
Fourth block pooling layer	43.8	46.8	45.3
Final pooling layer	41.0	46.2	43.4

As Table 5.2 shows, the F-score value for the best performing system, the increased fourth block system, is around 20% lower than the baseline system in relative terms. The per-class performance will be analysed in detail and compared to the baseline system to understand why this system is performing poorly.

The principal components early fusion system performs worse than the baseline on almost every class Figure 5.2. However, it does perform better on classes that the full prediction early fusion system performed worse than the baseline on, such as ‘car alarm’ and ‘reversing beeps’. These events are linked only to audio domain and do not have a

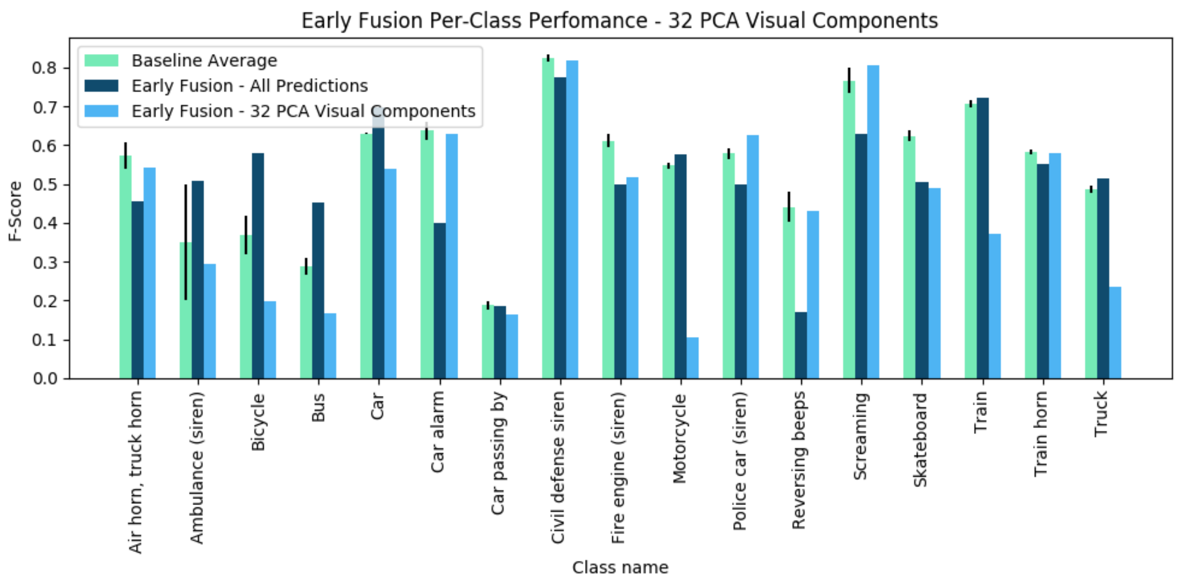

Figure 5.2: Per-class performance for the early fusion reduced Inception predictions system

representation in the visual domain, which means that the principal components early fusion system is learning to ignore the visual information for these audio-only events.

On the other hand, classes that the full prediction early fusion system outperformed the baseline system on such as ‘bicycle’, ‘bus’, and ‘car’, which are all events that are tied with objects in the visual domain, performed much worse than both the initial early fusion system and surprisingly, the baseline. By comparing the confusion matrices of the two systems Figure 5.3 and Figure 5.4, it can explain why the principal components system does not perform as well. The confusion between the visually represent classes, those that the visual feature extractor would be able to detect such as ‘bicycle’, ‘bus’ and ‘car’, is much higher in the principal components early fusion system than the full prediction early fusion system. This shows that while 32 principal components were chosen since it represents the fact that visual data gives less information to the audio tagging problem than the sound itself, not enough principal components were used to sufficiently capture the visual data. While it may be possible to find a balance between the full prediction vector and 32 principal components of that vector by using a higher number of principal components, it will likely not achieve a greatly improved result.

5.3 Summary

While these results have shown that early fusion did not improve the performance of the baseline audio tagging system, it cannot be said that early fusion is not useful. The full predictions early fusion system shows that some audio classes can be improved using audio and visual data with early fusion techniques, specifically those that have object representations in the visual domain. However, using early fusion techniques can harm the performance of event classes that do not have a representation in the visual

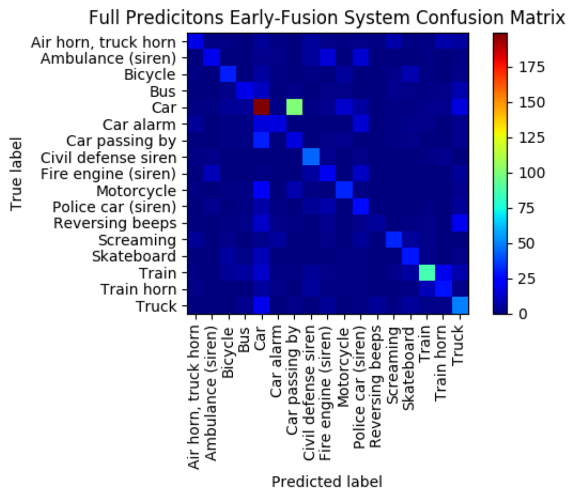

Figure 5.3: Confusion matrix for the full predictions early fusion system

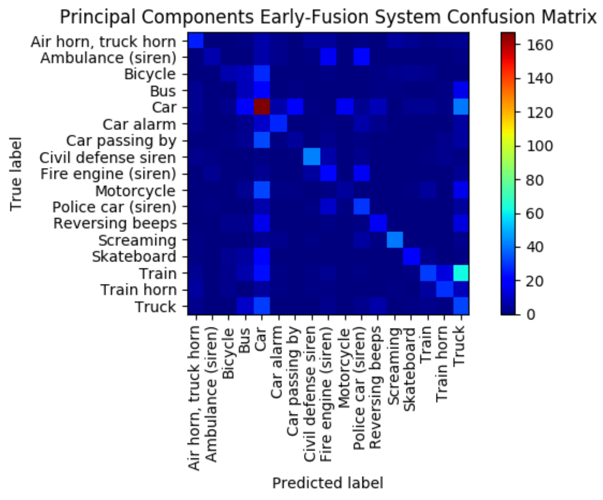

Figure 5.4: Confusion matrix for the reduced predictions early fusion system

domain. In order for an audio visual sound event detection system to be advantageous, the visual information should be ignored for classes which the visual information harms the performance of.

The second early fusion experiment was to reduce the dimensionality of the visual features using principal component analysis so that their dimensionality matched the amount of information they provided in comparison to the audio features. The results of this experiment were mostly the opposite of the full predictions early fusion experiment, with the performance of the visually represented classes decreasing and the performance of the audio-only classes being around the same as the baseline system. The reason for this is that the number of chosen components was not enough to represent the variation in the Inception predictions, which meant that objects were not distinct, increasing the confusion between visual objects.

The full predictions early fusion system has shown some benefits, and there is the possibility of further modifying the technique to bring some overall performance improvements. The biggest drawback of these early fusion experiments was that the visual predictions had to be repeated in the time domain to match the audio features. Further experiments could be carried out by modifying where the feature fusion takes place, so that the predictions do not need to be repeated. Further, feeding the predictions through a CNN is also not well motivated, since the predictions do not form an 'image' like the audio spectrograms.

6. Experiment 2: Late Fusion of Audio and Visual Features

Instead of early fusion of the features by combing both the audio spectrogram and the Inception predictions before it is passed into the baseline classifier, these experiments will be focused on late fusion techniques.

The reason for exploring late fusion is because the early fusion results showed that visual information can improve the performance of detecting sound event classes that have a representation in the visual domain. Yet, early fusion severely reduces the performance of sound event classes that do not have a representation in the visual domain, as the network does not learn to ignore the visual data in those cases. Another reason is that the Inception predictions do not change over time, because they were taken from a single frame and that prediction is repeated in the time domain to match the audio features. This is counter-intuitive for the baseline network as the convolutional aspect tries to learn change over time, but cannot learn any change from the visual predictions.

By using late fusion instead, the audio baseline system will learn from just the audio data and not have to fit to multi-modal data, which should mean that it learns a better representation of the audio data. Further, the way in which the modalities are combined will mean that unreliable visual information can be discarded, while keeping the reliable information to help improve the classes that have a representation in the visual domain.

There are many ways of implementing late fusion with these two systems. One method is to adjust the Inception network to output probabilities for the DCASE classes, that is, change the output of the network to only have 17 outputs that map to the DCASE classes. Once it is adjusted, the outputs of the Inception system can be combined with the output of the audio system using a linear method. Otherwise, the full Inception prediction vector can be concatenated with the predictions from the audio tagging baseline and fed through a third network to get the final predictions.

6.1 Adjusting the Inception Network

In order for the Inception network to output the probabilities for each of the 17 DCASE labels for an image, the network needs to be adjusted. There are two options for adjusting the network: transfer learning or fine-tuning.

Transfer learning is when the output of the layer behind the fully connected layers is taken and a classifier is trained on top of it to draw decision boundaries for the new classes. This can be a useful option on a small dataset. Fine-tuning is where a classifier is placed on top of the Inception network and trained, like transfer leaning, and also training (fine-tuning) the top few layers of the Inception network. By keeping the early layers of the network frozen so that their weights are the same as the trained network, the features that they learned to extract by being trained on the large ImageNet dataset are still applied to the new dataset. Fine-tuning generally results in better performance than transfer learning when the dataset is large enough, as the later layers which extract high-level features become tuned to the new dataset.

As the dataset for this project is fairly large with more than 40,000 videos, fine-tuning is chosen to be the method to adjust the Inception network to the current dataset. However, there is an issue in that the dataset is very imbalanced. This means fine-tuning the network could overfit on the classes that do not have many examples in the training dataset, such as 'reversing beeps' and 'train horn', and not generalise well to the testing and evaluation datasets.

6.1.1 Setup

There are two steps to fine-tuning: First a classifier is placed on top of the Inception network and trained, while the layers of the Inception network are frozen. This is the same as transfer learning, and must be done so that the randomly initialised weights of the new layers do not remove the features previously learnt by the Inception network. Secondly, some of the later layers are unfrozen, in this case the final two inception blocks, and trained on the new dataset along with the new top layers so they extract features specific to this dataset.

The output layer of the inception network which has 1000 outputs will be replaced with an output layer containing 17 outputs with each corresponding to a label of the DCASE challenge. The structure of the new classifier is described in Table 6.1. The first layer of the fine-tuning network is GlobalAveragePooling2D, as the data needs to be 2-dimensional and the output of the Inception network is 4-dimensional. Therefore this first layer reduces the dimensionality of the data by using average pooling on all the spatial dimensions until they are one. An alternative would be to use flatten, but that results in a larger dense layer and may cause the network to overfit. BatchNormalisation is used because it allows the network to converge faster, reduces sensitivity to the initial weights and provides regularisation to the network [54]. It does this by normalising the input of each layer so that the mean is zero and standard deviation is one. Dropout layers are used because after first training the fine-tuning network, it was clear that the network was over-fitting on the training dataset since the training loss continually decreased while the testing loss increased. Dropout layers have been proven to help avoid over-fitting by randomly dropping a fraction of the output neurons in a layer, stopping neurons in a layer co-adapting [55]. The dropout rate is set to a high value, 80% since it was found to avoid over-fitting better than lower values.

The output layer contains 17 neurons, which map to the DCASE classes, and uses a sigmoid function for the activation. The reason for using sigmoid is that, because the probability distribution for the softmax function sums to 1, the prediction probabilities cannot be independent. However, since it is possible that the audio clips have multiple labels, the probabilities should be independent as different events could occur in a single clip. In contrast, in the sigmoid function the prediction probabilities are independent, and therefore it is the chosen activation function for the output layer.

Softmax Probability Distribution:

$$P(c_j|x_i) = \frac{\exp(z_j)}{\sum_{k=0}^{N} \exp(z_k)}$$

Where z_j is the probability of example x_i belonging to class c_j and N is the number of classes

Sigmoid Probability Distribution:

$$P(c_j|x_i) = \frac{1}{1 + \exp(-z_j)}$$

Where z_j is the probability of example x_i belong to class c_j

Table 6.1: Fine-tuning Architecture

Type	**Notes**	**Input size**
GlobalAveragePooling2D		1x1x1x2048
BatchNormalisation		1x2048
Dropout	0.8	1x2048
Dense	ReLU activation	1x1024
BatchNormalisation		1x1024
Dropout	0.8	1x1024
Dense	Sigmoid activation	1x17

The chosen loss function is binary cross entropy, since it computes the loss for the classified labels independently, which along with the sigmoid activation function, is what is needed for this multi-label problem. The loss function needs to be weighted as some of the classes have many more training examples than others, so the bias towards these classes needs to be removed. The baseline system adjusts the data in each batch so that the number of examples for each class is balanced, however implementing the same mechanism for the frames is difficult due to some limitations in the ImageDataGenerator function provided by Keras. Instead, the fit_generator method in Keras takes a class_weight parameter, which is used to weight the classes in the loss function. The class_weight dictionary is computed using the compute_class_weight function in the sci_kit learn library, by counting the number of examples for each class in the training data.

Two different optimisers were used in training the network. The first part of training - transfer learning, used the RMSprop optimiser, which the Inception network uses while originally training [2]. RMSprop is an adaptive optimisation method that changes the learning rate dynamically. For the second part - fine-tuning, a non-adaptive learning rate optimiser such as SGD is recommended to ensure that the parameter updates remain small, so as to not remove the features the Inception network has previously learnt from the larger ImageNet dataset. Therefore, SGD is the optimiser chosen for fine-tuning the system.

During training, the images will also undergo random augmentation, e.g. rotation and zoom, which means the dataset will not see the same image twice, reducing over-fitting and increasing the ability of the network to generalise. This is implemented via a function provided by Keras - ImageDataGenerator. ImageDataGenerator provides a flow_from_directory function to load the images for each batch dynamically, so that the whole dataset does not have to be in memory. However, it is designed just for multi-class problems and not multi-label problems so that for each example, only one target label is given. It was modified so that for each example, the target vector is one-hot encoded. One-hot encoding means the target vector is the length of the number of classes in the problem, and the value is set to 1 for the classes the example contains, and the value is set to 0 for all the other classes.

6.1.2 Results

To evaluate the performance of fine-tuning the Inception network, the evaluation frames can be predicted by the network and evaluated using the DCASE evaluation metrics.

Table 6.2: Inception fine-tuned performance

Metric	**Precision**	**Recall**	**F1 Score**
Micro-averaged (DCASE metric)	16.9	17.9	17.3
Macro-averaged	5.46	8.0	6.49

The fine-tuning system gives an micro-averaged F-score of around 17% - Table 6.2, which seems impressive considering that the system is trying to predict audio events from a single still image. However if we look at the macro-averaged F-score, which averages the F-score over each of the classes instead of counting all the TPs, FPs etc. together, it is much lower at 6%. As discussed in subsection 2.1.2, macro-averaging biases the metric towards classes with most examples, while micro-averaging biases the metric towards the classes towards the classes with least examples [19]. Macro-averaging is not used by the DCASE challenge, but it reveals that the fine-tuned system is biased towards classes with most examples as the macro-averaged F-score is much higher then the micro-averaged F-score. If we compare the micro-averaged and macro-averaged F-score performance for the baseline system Table 6.3, we can see that the micro-averaged and macro-averaged F-score is very similar, therefore the baseline is not biased towards under or over-represented classes in the dataset.

Table 6.3: Baseline macro and micro-averaged F-score

Metric	**Precision**	**Recall**	**F1 Score**
Micro-averaged (DCASE metric)	56.0	58.4	57.2
Macro-averaged	56.4	56.0	56.2

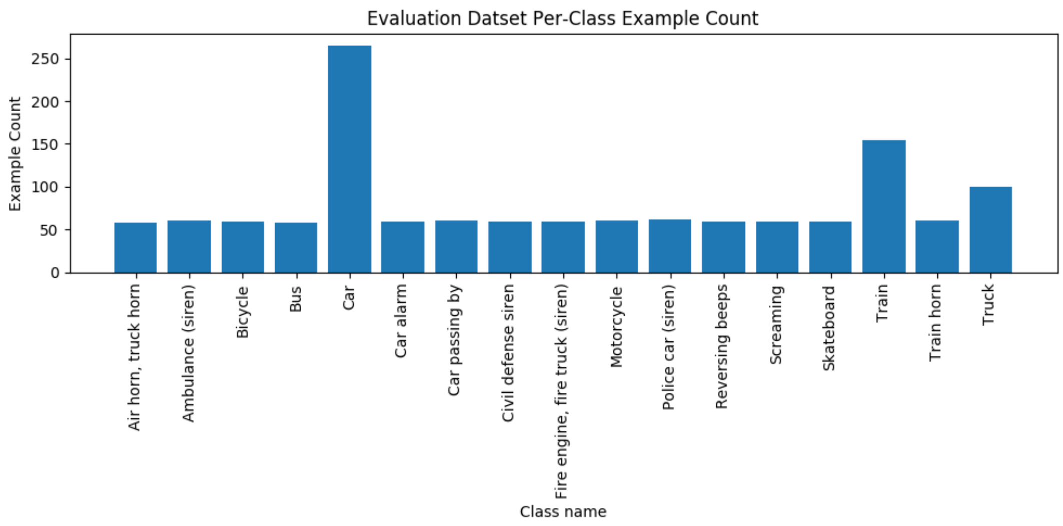

Figure 6.1: The count for the number of examples of each class in the evaluation dataset

The reason is that the ‘car’ class, which represents most of the evaluation dataset - Figure 6.1, is performing well while most of the other classes are under-performing Figure 6.2. While the idea behind using a weighted loss function was to reduce the impact the imbalanced dataset would have on the performance of the network, it is clear that the network is still better at recognising the ‘car’ event than any other. However, since the training data only used a single frame from the middle of the video, a solution could exist in extracting frames from other parts of the video for under-represented classes.

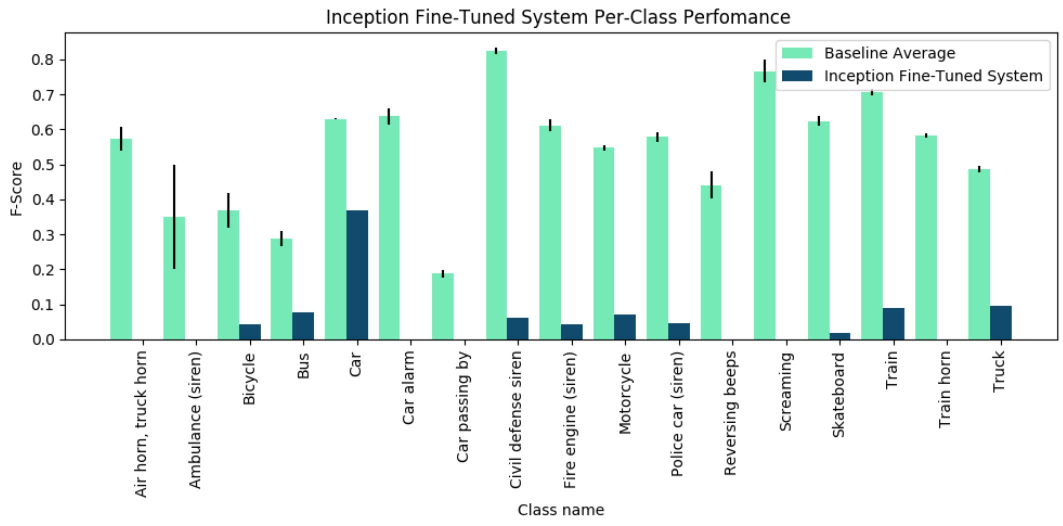

Figure 6.2: Per-class performance of the fine-tuned Inception network

The fine-tuned network is not able to recognise any sound events that do not have a representation in the visual domain. Looking at the per-class performance Figure 6.2, it was not able to identify any of the ‘air horn, truck horn’, ‘car alarm’ or ‘screaming’ sound events among others. While these events may have some representation in the visual domain, it is not a strong enough connection for the visual system to pick up. For instance, it might be expected that the visual system could predict the ‘air horn, truck horn’ class whenever it sees a truck object. However there is also a ‘truck’ sound event class, for which the visual only system does predict events. This can explain why the visual system cannot predict the ‘air horn, truck horn’ class, since there is no difference between a ‘truck horn’ sound and ‘truck’ sound in the visual domain. The same reasoning explains why the network is not able to tell the difference between the ‘car’, ‘car alarm’ and ‘car passing by’ classes, and the ‘train’ and ‘train horn’ classes.

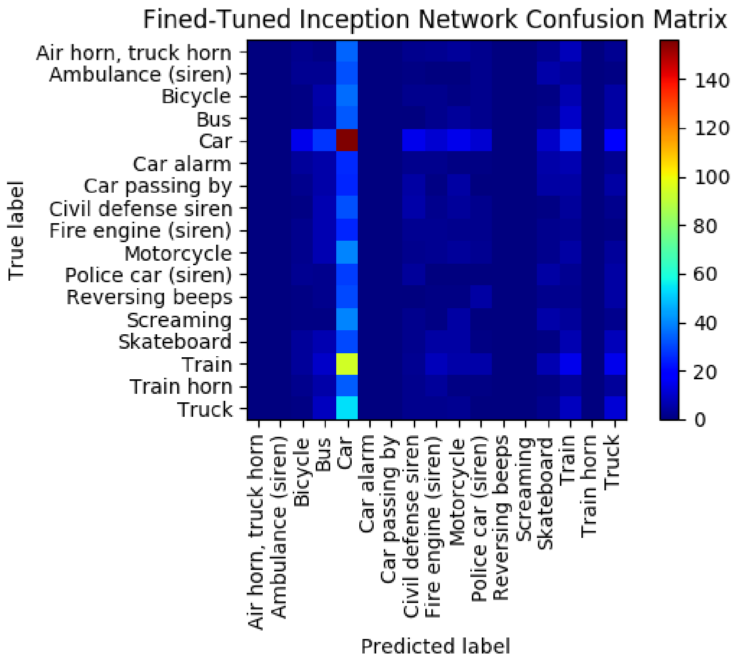

Figure 6.3: Confusion matrix for the fine-tuned Inception network

However, perhaps the most surprising results from fine-tuning the network is that performance has been less than expected on the ‘bicycle’, ‘bus’, ‘motorcycle’, ‘train’ and ‘truck’ classes, as using the early fusion technique resulted in an increase in performance over the baseline for these classes - Figure 5.1. Looking at the confusion matrix - Figure 6.3, we can see that most mistakes are made because the network thinks it is seeing a car. If we look at some frames from the ‘bicycle’ classes - Figure 6.4, the reason for some of this confusion becomes clearer. As well as dealing with a highly unbalanced dataset, many of the frames of the other classes contain cars even though the labels may not reflect this. Further, since the dataset is weakly labelled, it means that the precise location of the sound emitting objects is not known to the fine-tuning network. This means that if there is a car in the frame but it does not emit any sound, then the example will not be classified as a ‘car’, but the fine-tuning network will ‘see’ the car and think that it is the object that corresponds to the output label. Therefore, the network becomes very good at finding cars, and since the dataset contains many more ‘car’ examples than anything else, the result is that any time the network sees a car in an examples it classifies that example as the ‘car’ class.

Figure 6.4: Two frames from the dataset with the label ‘Bicycle’ [13]

6.2 Decision Fusion

Despite the poor performance of the fine-tuned Inception network, the posterior probabilities from the audio-only baseline and visual-only baseline will be combined in a method called decision fusion, to give a baseline for the late fusion experiments. Decision fusion combines the probabilities from different classifiers by running the examples through each classifier and combining the posterior probabilities by some linear combination. The problem with this method is that classes that the visual system is unreliable at predicting, such as classes without a visual representation like ‘reversing beeps’, have the same weighting as the classes that the visual system is reliable at predicting like the ‘car’ class.

6.2.1 Setup

There will be three linear combinations used in this experiment. Multiplying the posterior probabilities of each of the classes from both of the systems, taking the maximum probability for each of the classes from both systems, and lastly, averaging the probabilities from the two systems. When averaging the probabilities from the two systems, the systems can be weighted, so that the more reliable audio system has higher influence over the final prediction than the visual system. The weight for the audio baseline system is set using variable α, then the weight for the fine-tuned visual system is calculated by $1 - \alpha$. I.e. when $\alpha = 1.0$ then the output is the audio baseline predictions, and when $\alpha = 0.0$ then the output is the fine-tuned visual system predictions.

6.2.2 Results

By weighting the systems when multiplying the posterior probabilities, the influence each system has on the final prediction changes. As Figure 6.5 shows, because the audio baseline is much more reliable than the fine-tuned Inception system, the higher the weighting is towards the audio baseline, the higher the F-score after multiplying the predictions.

Table 6.4: Individual performance of the two systems to be combined using decision fusion

System	Precision	Recall	F1 Score
Baseline	$55.7 \pm 2.2\%$	$58.1 \pm 1.9\%$	$56.9 \pm 2.0\%$
Inception Fine-tuned	16.9	17.9	17.3

Table 6.5: Decision fusion overall performance

Combination Method	Precision	Recall	F1 Score
Average	42.3	42.8	42.5
Maximum	34.3	59.6	43.6
Multiplication ($\alpha = 0.93$)	56.8	54.9	55.8

The graph shows a slight increase in performance when $\alpha = 0.93$, which is the reported result in Table 6.5.

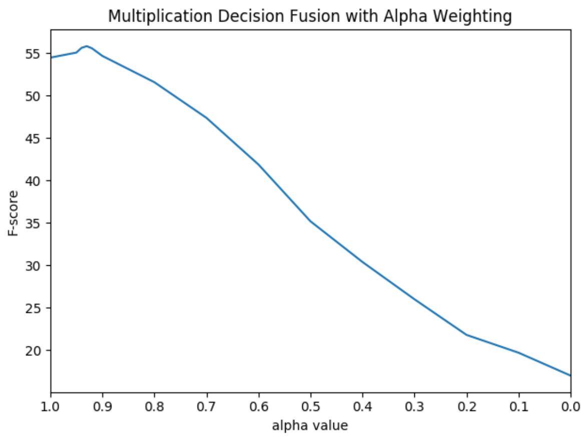

Figure 6.5: Effect of weighting on multiplication decision fusion F-score

Even though the fine-tuned Inception network performs poorly because of the previously mentioned problems, decision fusion resulted in a slight increase in F-score compared to the baseline system, although it was not a significant improvement. This shows that if the issues with fine-tuning the network can be resolved, then even the simplest late fusion method could result in a significant increase in performance over the audio baseline.

6.3 Late Fusion of Inception Predictions and Baseline Predictions

Fine-tuning the Inception network to output probabilities for the DCASE classes has not been beneficial. Since combining the full Inception prediction vector with the audio features in the early fusion experiments resulted in improved performance for the visually represented classes, this experiment will integrate the Inception predictions with the predictions from the audio tagging baseline using a third network. The early fusion network was able to extract some useful information about the sound events with representation in the visual domain, but also decreased the performance of events without representation in the visual domain. The idea behind this experiment is that because the network will not be convolutional, then it should learn to ignore the Inception predictions for sound-only classes, while being able to integrate the information for sound events that are multi-modal.

6.3.1 Setup

The input vector will be the audio baseline predictions concatenated with the Inception predictions vector to give a 1017 dimensional vector. This will then be fed through the integration network to output a 17 dimensional vector mapping to the audio DCASE classes.

Most of the design principals of this network are carried from the Inception fine-tuning network. The loss function will continue to be binary cross entropy with class weighting, and the activation function for the output layer will be sigmoid for the same reasons. Each layer of the network will consist of a dense layer using either a relu or tanh activation function, followed by batch normalisation and a 50% dropout. The effects of using relu or tanh activation functions will be compared along will using different depths of networks.

$$\text{Tanh activation function:} \quad \tanh(a) = \frac{\sinh(a)}{\cosh(a)}$$

The adam optimiser is used since it is robust and converges quicker than conventional gradient descent optimisers [48]. The batch size was set to 64, which was small enough to fit into memory and performed better than smaller batch sizes. The two different types of network will be named deep and wide, consisting of four and two layers respectively. The deep network consists of layers that begin at 1024 neurons, then layers of 512, 256, 128 neurons, followed by a 17 neuron layer for the output. Whereas the wide network consists of a 2048 neuron layer, followed by a 1024 neuron layer and finally the 17 neuron output layer.

6.3.2 Results

Table 6.6: Performance comparison of Inception predictions and baseline predictions late fusion systems

System	**Precision**	**Recall**	**F1 Score**
Deep network with ReLU activation	57.4	56.2	56.8
Deep network with tanh activation	57.4	56.2	56.8
Wide network with ReLU activation	55.3	54.6	55.0
Wide network with tanh activation	56.5	56.8	56.7

As the system performance comparison shows - Table 6.6, the different network architectures are closely matched in terms of performance, and all are in the range of the baseline system - Table 3.1. The per-class performance of the two best systems, the deep networks using ReLU and tanh activation functions, will be compared with the baseline and the best early fusion system to examine how the type of fusion affects the performance for the different classes.

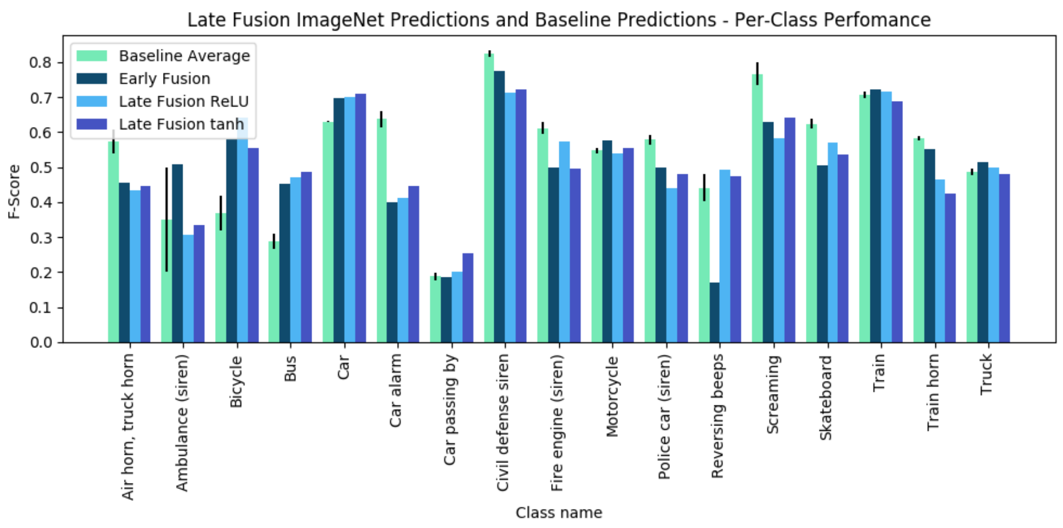

Figure 6.6: Comparison of late fusion systems per-class performance with the early fusion and baseline systems

Looking at the per-class performance - Figure 6.6, we can see that again like the early fusion system, the performance for classes that have a representation in this visual domain such as ‘bicycle’ and ‘bus’, has improved in comparison to the baseline system. Unlike the early fusion system, the late fusion systems have mostly learned to ignore the visual features for events that only have a representation in the audio domain. For example, the performance of the ‘reversing beeps’ class in the late fusion systems is much closer to the baseline’s performance than the early fusion system. Although it seems as though the late fusion system is performing worse on the ‘ambulance siren’ class than the early fusion system, both the early and late fusion systems are within the variance of the baseline system and it is therefore not a significant difference. A class in which the early fusion system outperforms the late fusion systems is the ‘train horn’ class. If we look at the confusion matrices of the late fusion systems Figure 6.7 and Figure 6.8 and the full predictions early fusion system Figure 5.3, we can see that the late fusion systems misclassify the ‘train horn’ examples as other audio-only classes more often than the early fusion system. Between the two late fusion systems, there is not much difference in performance for most of the classes. The ReLU system gives slightly improved performances on the visually represented classes such as ‘bicycle’ and ‘skateboard’. Whereas, the tanh system gives slightly improved results on the audio-based classes such as ‘car alarm’ and ‘police car siren’. There is an exception to this rule with the ‘fire engine siren’ class where the ReLU system performs close to the baseline performance.

Surprisingly, the per-class results show a slight increase for ‘car passing by’ class. This is unexpected as the Inception predictions do not include any temporal information, therefore combining them with the baseline predictions should not increase the performance

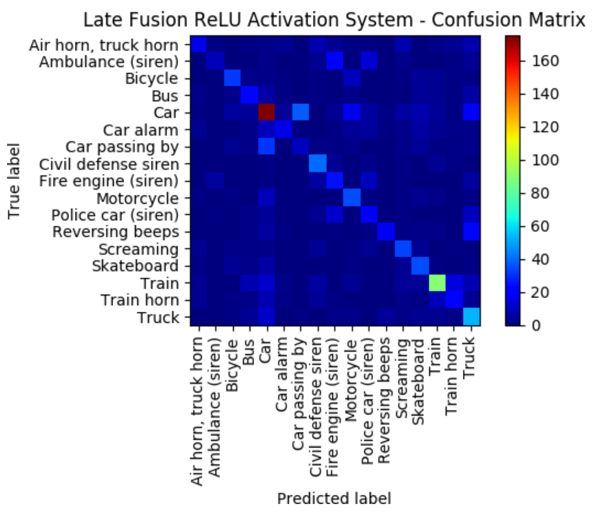

Figure 6.7: Confusion matrix for the ReLU late fusion system

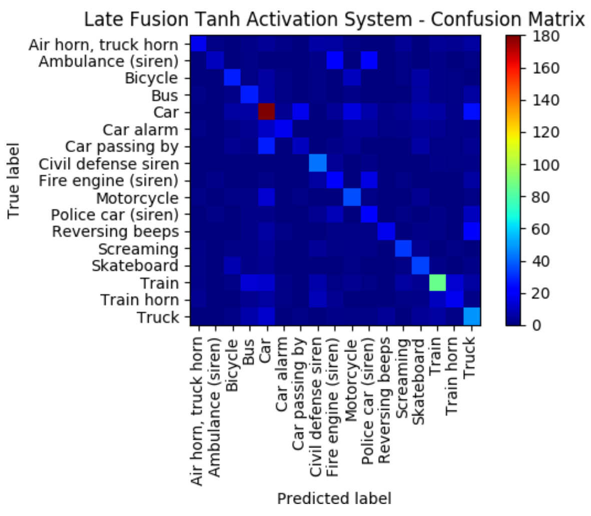

Figure 6.8: Confusion matrix for the tanh late fusion system

of this class. Looking at the precision and recall metrics for the 'car passing by' class - Table 6.7 from each of the systems, it shows that the recall for the class was higher in the early fusion system than the baseline system. Yet the precision was decreased, giving an overall F-score similar to the baseline. The higher recall is caused from the system using the visual predictions of a car to increase the number of examples classified as 'car passing by', but this also decreases precision as examples for the 'car' class are classified as the 'car passing by' also. The late-fusion tanh system increases the precision on the 'car passing by', which decreases confusion between the 'car' and 'car passing by' classes. By looking at some images, we can try and understand how the network is separating the two classes. The examples that the system managed to correctly classify as the 'car passing by' class - Figure 6.9, all have something in common with racing. Whereas the examples that the system incorrectly did not classify as 'car passing by' - Figure 6.10, are all ordinary scenes and have nothing to separate them from the 'car' class.

Table 6.7: Car passing by class performance system comparison

System	**Precision**	**Recall**	**F1 Score**
Baseline	0.173	0.204	0.187
Early Fusion	0.129	0.34	0.187
ReLU Late Fusion	0.204	0.2	0.202
Tanh Late Fusion	0.345	0.2	0.253

6.4 Summary

The late fusion experiments have ultimately not improved the overall performance of the baseline system, yet it has proved that audio-visual data can be useful with certain sound event classes. The early fusion experiments also showed some improvement, but the negative effects on the audio-specific classes made it hard to motivate the use of an early fusion technique for this task. Most of the problems can be attributed to the dataset

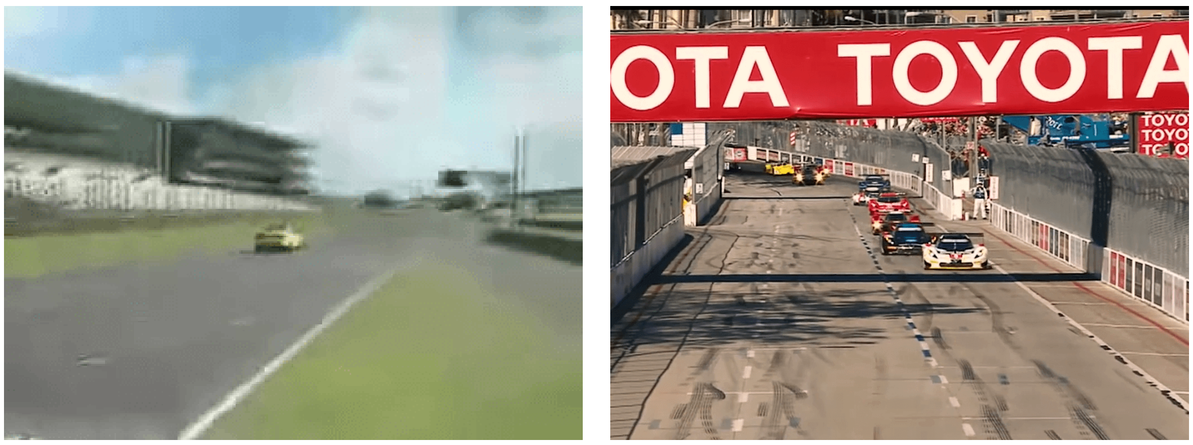

Figure 6.9: Examples classified correctly as 'car passing by' by the tanh late fusion system [13]

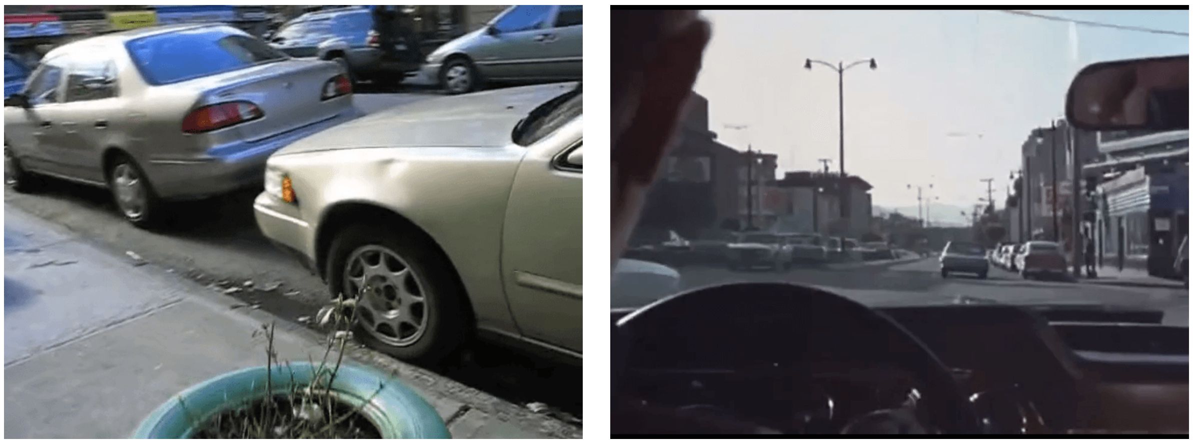

Figure 6.10: Examples incorrectly not classified as 'car passing by' by the tanh late fusion system [13]

used in this project, rather than the idea of using visual information to enhance a audio tagging system.

Although it was expected that the fine-tuned network would not work for classes without visual representation, it was disappointing that the network did not mange to capture the classes closely tied to a visual representation either. This can be attributed to the dataset for two reasons. Firstly, the number of car examples compared to all of the other classes already meant that problem was biased towards the 'car' class and had to be overcome by weighting the loss function. Secondly, and more importantly, the abundance of car objects in frames from other classes and the lack of positional information due to the weakly labelled dataset exacerbated the imbalanced dataset. These two reasons meant that the fine-tuned network could only recognise cars, and a small number of other objects in uncluttered frames.

Late fusion has shown that it can improve on some of the problems that early fusion created, such as decreased performance on audio classes that do not have a representation in the visual domain, and possibly improve on those classes due to the typical environments they take place in. This use of environmental information to help improve audio representations is the idea behind the SoundNet network [31], and confirms the amount of information that the visual domain contains about the sound domain.

However, due to the prevalence of cars in the video frames and the unknown exact location of the audio event, meaning that the correct object may not be in the video frame, the overall performance of the baseline audio tagging system was not improved by combining the visual information in a late fusion technique.

7. Conclusion

This project aimed to leverage the visual information to improve the performance of an existing audio-only sound event tagging system by using multi-modal techniques. Multi-modal learning is a promising technique that can be used to improve the performance of systems in many ways. It can reduce the impact of noise, allow systems to generalise better when using smaller datasets, and generally enhance the performance of previously single modality problems. The sound events dataset used to train the baseline system is a subset of a YouTube based dataset, which made using the visual information a clear next step. The dataset is weakly labelled, which means that for each 10 second clip in the dataset, the labels for each sound event that occurred are given but not the exact locations of the events.

The experiments for the project were split into two groups, early fusion and late fusion, corresponding to the type of multi-modal fusion they used. Early fusion combined the features from each modality into a single representation and in late fusion each modality had its own classifier, the audio baseline and the Inception network, and the decisions from both were combined in different ways. Both experiments used the Inception network, an image object detection network trained on the ImageNet dataset, to extract information from the visual domain by classifying the objects in each frame.

7.1 Findings

Two early fusion techniques were tested, concatenating the full Inception prediction vector with the audio features, and reducing the Inception predictions using principal component analysis before concatenating them. Both techniques did result in an increase in the overall system F-score. However, using the full predictions vector from the Inception network did increase the performance of the visually represented classes such as ‘bicycle’, but also decreased the performance of the classes with no visual representation, such as ‘reversing beeps’. These results show that the visual domain does indeed contain useful information about the audio domain for audio-based classification tasks like audio tagging. Reducing the dimensionality of the predictions before concatenation resulted in a decrease in overall performance and in the performance of the visually represented classes.

Two late fusion techniques were also tested. There first late fusion technique was to fine-tune the Inception network so that the output was the 17 DCASE classes and then combine the posterior probabilities using linear methods. This method was unsuccessful due to the imbalanced dataset and its lack of strong labels, meaning the fine-tuned network was not able to learn good representations of the objects other than cars. The second method was to combine the Inception ImageNet predictions with the audio baseline predictions and train a third network on this data. This system again did not improve the overall performance of the baseline, but it did provide some interesting results. The visual information was taken into account for visually represented objects in the same way as the early fusion systems, but unlike those systems, the visual information was ignored when it was unhelpful. It also improved a temporal audio event, 'car passing by', without extra temporal information, by leveraging the environment information from the visual information, in the same way as the SoundNet system did for audio scene classification.

The reason these experiments did not improve the performance of the existing audio tagging system is due to the imbalanced dataset and the fact that it is weakly labelled. The extraction of frames and consequent visual feature extraction did not take into account the weak labels, meaning it was assumed that the sound emitting object was in the extracted middle frame, which is clearly not the case for all of the videos.

While this project has not found any improvement in overall performance by integrating visual information into an existing audio-only tagging system, it cannot be said that visual information is not useful for audio tagging and sound event detection. Both the early and late fusion techniques resulted in improvements in the visually represented sound events such as 'bike' and 'bus', and the late fusion system showed visual information can aid long-duration sound events e.g. 'car passing by'.

7.2 Further Work

As stated, the reason these experiments were unsuccessful at improving the performance of the audio tagging baseline system using the visual information is because of the imbalanced dataset and the weak labelling of examples. Therefore, further work could be carried out to minimise the effect of weak labels on the visual feature extraction, by using a technique called multiple-instance learning. If the input frame to the visual feature extractor was a frame which contained the object that emitted the sound event, then the predictions would be more accurate, which might enhance the performance of the late fusion multi-modal system. Further, with this frame issue solved, the Inception network could again be fine-tuned to see the performance of a visual-only system on this audio tagging problem. With the newly fine-tuned network, the posterior probabilities from both classifiers could be combined in a non-linear method by using a neural network.

A study, using the same dataset as this project, that solved some of the problems that arose in this project was published shortly after these experiments were concluded [56]. The system localises the sound events in the audio and visual domains separately, so if the object that emits the sound does not appear in the video until after the sound event, then the correct visual frame is used to extract features from. The localisation method also locates the position of the sound-emitting object within the frame, giving a bounding box for the object. This localisation address two of the major flaws in this project: the choice of which frame(s) to extract from the videos, which in this project was chosen to be the middle frame, and the fine-tuning of the Inception network without positional information. This localisation method could be integrated into the visual feature extractor, so that the correct frame is classified by the ImageNet network before integrating into the existing baseline using the late fusion technique.

Finally, the dataset could be expanded outside of the classes that were provided for the DCASE challenge. The AudioSet dataset contains 632 classes of events, all of which are collated from YouTube videos meaning there is easy access to a large audio-visual dataset. The DCASE dataset was not a very diverse dataset, mainly containing examples that occurred in a street scene. Therefore the techniques investigated in this project could be applied to a different dataset to see the effect of using an audio-visual multi-modal approach on that dataset. Since the final late fusion experiment showed that the network was using visual environmental information to differentiate between 'car' and 'car passing by', a multi-modal approach may have more of an effect on a dataset that contains sound events that occur in different environments.

Bibliography

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[2] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *ArXiv preprint*, 2015.

[3] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 2015.

[4] P. K. Atrey, N. C. Maddage, and M. S. Kankanhalli, "Audio Based Event Detection for Multimedia Surveillance," in *IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, 2006.

[5] D. Zhang and D. Ellis, "Detecting sound events in basketball video archive," Dept. Electronic Eng., Columbia Univ., New York, Tech. Rep., 2001.

[6] J. Portêlo, M. Bugalho, I. Trancoso, J. Neto, A. Abad, and A. Serralheiro, "Non-speech audio event detection," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2009, pp. 1973–1976.

[7] Y.-T. Peng, C.-Y. Lin, M.-T. Sun, and K.-C. Tsai, "Healthcare audio event classification using hidden markov models and hierarchical hidden markov models," in *IEEE International Conference on Multimedia and Expo*, 2009, pp. 1218–1221.

[8] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "SoundSense: scalable sound sensing for people-centric applications on mobile phones," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*, 2009, pp. 165–178.

[9] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal Deep Learning," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 689–696.

[10] K. Noda, Y. Yamaguchi, K. Nakadai, H. G. Okuno, T. Ogata, K. Noda, T. Ogata, Y. Yamaguchi, H. G. Okuno, and K. Nakadai, "Audio-visual speech recognition using deep learning," *Applied Intelligence*, vol. 42, pp. 722–737, 2015.

[11] J. Huang and B. Kingsbury, "Audio-visual deep learning for noise robust speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 7596–7599.

[12] J. F. Guitarte Pérez, A. F. Frangi, E. L. Solano, and K. Lukas, “Lip Reading for Robust Speech Recognition on Embedded Devices,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005, pp. 473–476.

[13] J. F. Gemmeke, D. P. W Ellis, D. Freedman, A. Jansen, W. Lawrence, R. Channing Moore, M. Plakal, and M. Ritter, “Audio set: An ontology and human-labeled dataset for audio events,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 776–780.

[14] D. Stowell and D. Clayton, “Acoustic event detection for multiple overlapping similar sources,” in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*, 2015, pp. 1–5.

[15] J. Salamon, B. Mcfee, P. Li, and J. P. Bello, “Multiple instance learning for sound event detection,” *Technical report, DCASE2017 Challenge*, 2017. [Online]. Available: `http://www.cs.tut.fi/sgn/arg/dcase2017/documents/challenge_technical_reports/DCASE2017_Salamon_192.pdf`.

[16] *Weak Supervision - Overview.* [Online]. Available: `http://hazyresearch.github.io/snorkel/blog/weak_supervision.html`.

[17] A. Mesaros, T. Heittola, and T. Virtanen, “Metrics for Polyphonic Sound Event Detection,” *Applied Sciences*, vol. 6, p. 162, 2016.

[18] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, “DCASE 2017 challenge setup: Tasks, datasets and baseline system,” in *DCASE Workshop on Detection and Classification of Acoustic Scenes and Events*, 2017.

[19] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Information Processing and Management*, vol. 45, pp. 427–437, 2009.

[20] Krishan, *Deep Learning and Feature Engineering – From Data to Decisions*, 2015. [Online]. Available: `https://iksinc.wordpress.com/2015/12/18/feature-engineering-and-deep-learning/`.

[21] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European conference on computer vision*, 2014, pp. 818–833.

[22] Y. Xu, Q. Kong, W. Wang, and M. D. Plumbley, “Large-scale weakly supervised audio classification using gated convolutional neural network,” *ArXiv preprint*, 2017.

[23] S. Adavanne and T. Virtanen, “Sound event detection using weakly labeled dataset with stacked convolutional and recurrent neural network,” *ArXiv preprint*, 2017.

[24] J. Lee, J. Park, and J. Nam, “Combining multi-scale features using sample-level deep convolutional neural networks for weakly supervised sound event detection,” *Technical report, DCASE2017 Challenge*, 2017. [Online]. Available: `http://www.cs.tut.fi/sgn/arg/dcase2017/documents/challenge_technical_reports/DCASE2017_Lee_118.pdf`.

[25] *Mel Frequency Cepstral Coefficient (MFCC) tutorial.* [Online]. Available: `http://www.practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/`.

[26] X. Zhuang, X. Zhou, M. A. Hasegawa-Johnson, and T. S. Huang, “Real-world acoustic event detection,” *Pattern Recognition Letters*, vol. 31, pp. 1543–1551, 2010.

[27] E. Ç. Akır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, “Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, pp. 1291–1303, 2017.

[28] G. Parascandolo, H. Huttunen, and T. Virtanen, “Recurrent neural networks for polyphonic sound event detection in real life recordings,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016, pp. 6440–6444.

[29] H. Zhang, I. McLoughlin, and Y. Song, “Robust sound event recognition using convolutional neural networks,” in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 559–563.

[30] S. Hershey, S. Chaudhuri, D. P. W Ellis, J. F. Gemmeke, A. Jansen, R. Channing Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, “CNN architectures for large-scale audio classification,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 131–135.

[31] Y. Aytar, C. Vondrick, and A. Torralba, “SoundNet: Learning Sound Representations from Unlabeled Video,” *ArXiv preprint*, 2016.

[32] K. Mehrotra, S. Ranka, and C. K. Mohan, *Elements of artificial neural networks.* MIT Press, 1997.

[33] A. Karpathy, *CS231n Convolutional Neural Networks for Visual Recognition*, 2017. [Online]. Available: `http://cs231n.github.io/convolutional-networks/#overview`.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet : A Large - Scale Hierarchical Image Database,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[35] J. Deng, *Speed/Accuracy Trade-offs for Object Detection from Video Large Scale Visual Recognition Challenge 2017 (ILSVRC2017).* [Online]. Available: `http://image-net.org/challenges/talks_2017/Imagenet%202017%20VID.pdf`.

[36] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, “Robust real-time unusual event detection using multiple fixed-location monitors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 555–560, 2008.

[37] S. Ji, W. Xu, M. Yang, and K. Yu, “3D Convolutional Neural Networks for Human Action Recognition,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, pp. 221–231, 2013.

[38] Y. Ke, R. Sukthankar, and M. Hebert, “Efficient Visual Event Detection using Volumetric Features,” *IEEE International Conference on Computer Vision (ICCV)*, pp. 166–173, 2005.

[39] Y. Wang and F. Metze, “A Transfer Learning Based Feature Extractor for Polyphonic Sound Event Detection Using Connectionist Temporal Classification,” *Interspeech (ISCA)*, pp. 3097–3101, 2017.

[40] D. A. Sadlier and N. E. O’Connor, “Event detection in field sports video using audio-visual features and a support vector machine,” *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1225–1233, 2005.

[41] M. Cristani, M. Bicego, and V. Murino, "Audio-Visual Event Recognition in Surveillance Video Sequences," *IEEE Transactions on Multimedia*, vol. 9, pp. 257–267, 2007.

[42] C. G. M. Snoek, M. Worring, and A. W. M. Smeulders, "Early versus Late Fusion in Semantic Video Analysis," in *13th annual ACM international conference on Multimedia*, 2005, pp. 399–402.

[43] T. W. Lewis and D. M. W. Powers, "Audio-visual speech recognition using red exclusion and neural networks," *Australian Computer Society, Inc.*, vol. 24, pp. 149–156, 2002.

[44] R. Arandjelovi and A. Zisserman, "Look, Listen and Learn," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 609–617.

[45] ——, "Objects that Sound," *ArXiv preprint*, 2017.

[46] C. Hori, T. Hori, T.-Y. Lee, K. Sumi, J. R. Hershey, and T. K. Marks, "Attention-Based Multimodal Fusion for Video Description," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4203–4212.

[47] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Convolutional Gated Recurrent Neural Network Incorporating Spatial Features for Audio Tagging," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 3461–3466.

[48] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *ArXiv preprint*, 2014.

[49] A. Shah and R. Badlani, *Baseline for Task 4 Large-scale weakly supervised sound event detection for smart cars.* [Online]. Available: `https://github.com/ankitshah009/Task-4-Large-scale-weakly-supervised-sound-event-detection-for-smart-cars`.

[50] F. Chollet, *Keras Documentation.* [Online]. Available: `https://keras.io/`.

[51] jcjohnson, *CNN Benchmarks.* [Online]. Available: `https://github.com/jcjohnson/cnn-benchmarks`.

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[53] B. Castellano, *PySceneDetect.* [Online]. Available: `https://github.com/Breakthrough/PySceneDetect`.

[54] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training b y Reducing Internal Covariate Shift," *ArXiv preprint*, 2015.

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[56] S. Parekh, S. Essid, A. Ozerov, N. Q. K. Duong, P. Pérez, and G. Richard, "Weakly Supervised Representation Learning for Unsynchronized Audio-Visual Events," *ArXiv preprint*, 2018.