Sound Event Detection on Imbalanced Data using Spectral Entropy Active Learning

Moegamat Yusuf Ally

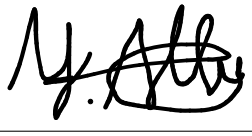
A dissertation submitted to the Faculty of Engineering and the Built Environment, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, October 2025

Declaration

I declare that this dissertation is my own, unaided work, except where otherwise acknowledged. It is being submitted for the degree of Master of Science in Engineering to the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination to any other university.

Signed this 1 day of October 2025



Moegamat Yusuf Ally

Abstract

Data imbalance is a detrimental factor in training Sound Event Detection (SED) systems. Long duration events are over-represented compared to short duration, minority events, and tend to overwhelm training. In addition, collecting and annotating sound event datasets is a laborious and expensive task. This work presents Spectral Entropy Active Learning (SEAL) to address both these problems. By exploiting the spectral characteristics of the audio, SEAL enables selection of short duration sounds which typically have low spectral entropy, thereby improving sampling distribution and increasing overall system performance. SEAL is compared against the state-of-the-art Asymmetric Focal Loss (AFL) and Unified Loss Function (ULF) methods, and it surpasses AFL F1-Micro performance by 0.31 percentage points, while requiring only 65% of the dataset to be labelled. Although ULF shows slightly better overall accuracy, it generates twice as many false positives as SEAL. Similarly, SEAL demonstrates better overall and classwise performance than reference Active Learning methods, such as Random Sampling, K-Medoids, and Least Confidence. The findings further suggest that for SED on imbalanced data, selection strategies that maximise the sampling distribution are more effective than uncertainty-based approaches. Future work will focus on improving SEAL's low labelling budget performance by integrating adaptive sample reweighting techniques, such as those used in AFL and ULF.

To my beloved mother, Sumaya...

Acknowledgements

I would like to express my gratitude to my supervisor, Professor Ling Cheng, who has provided support and guidance which extended well beyond the work presented here.

I wish to acknowledge the support and guidance provided by my father Sa-aydien, who has always encouraged life-long learning. Also I would like to thank my sister Fatima for providing financial assistance for Google Colab credits in order to carry out the experiments.

Contents

List of Figures

List of Tables

Acronyms

ADAM	Adaptive Moment Estimation
AFL	Asymmetric Focal Loss
AL	Active Learning
ANN	Artificial Neural Network
BiRNN	Bidirectional Recurrent Neural Network
BCE	Binary Cross-Entropy
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
DAFL	Deep Active Feature Learning
DCASE	IEEE Detection and Classification of Acoustic Scenes and Events
DNN	Deep Neural Network
ER	Error Rate
FL	Focal Loss
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
KMED	K-Medoids
LC	Least Confidence
LSTM	Long Short-Term Memory
ML	Machine Learning
PMF	Probability Mass Function

RAND	Random Sampling
RNN	Recurrent Neural Network
SE	Spectral Entropy
SEAL	Spectral Entropy Active Learning
SEC	Spectral Entropy Change
SED	Sound Event Detection
STFT	Short-Time Fourier Transform
ULF	Unified Loss Function

Publications

1. **M.Y. Ally**, L. Cheng, "Sound Event Detection on Imbalanced Data Using Spectral Entropy Active Learning", To be submitted to the IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2025

Chapter 1

Introduction

This chapter introduces the data imbalance problem in Sound Event Detection, and poses the question this work intends to answer: "*To what extent can Active Learning using Spectral Entropy as an uncertainty sampling selection metric, be used to reduce the effect of data imbalance in Sound Event Detection, as opposed to Asymmetric Focal Loss*". It also discusses the objectives and contributions of this work, and outlines the thesis structure.

1.1 Problem Statement

Audio data provides a rich source of information that is exploited by many pervasive sensing applications, such as personal activity monitoring, security surveillance, and health diagnosis. For instance, home intrusions can be detected from the sound of glass breaking, while domestic violence incidents can be characterised by distress-related vocalizations [1–3]. Developing these Sound Event Detection (SED) systems via supervised learning requires a large amount of labelled training data. Existing sound event datasets have an inherent data imbalance that is detrimental to SED accuracy. Figure 1.1 illustrates this imbalance in sound event distribution for the TUT Sound Events 2016/2017 and TUT Acoustic Scenes 2016/2017 combined datasets [4, 5]. Specifically, there are two levels of imbalance present: inter-class and intra-class imbalance. Long duration events like *Fan* and *Car* are over-represented compared to transient events like *Squeaking* and *Breathing* (inter-class imbalance), and this tends to result in lower SED model recall on short duration events. Moreover, the number of inactive frames, which are segments of training samples where there is no activity for a particular sound event, is exponentially higher than active frames (intra-class imbalance). By addressing the data imbalance, the model reduces bias towards

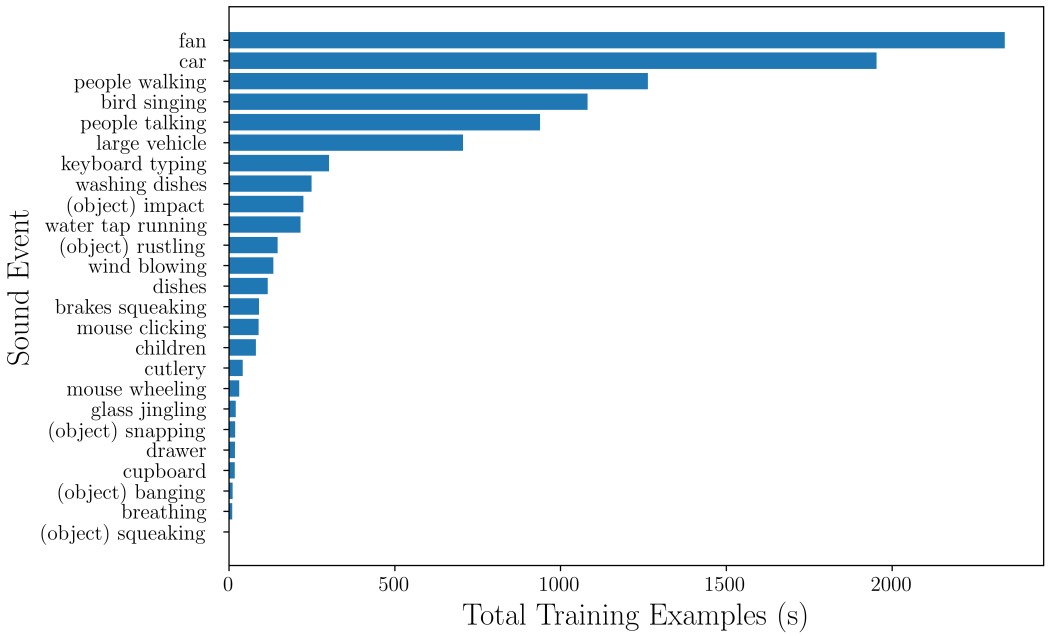


Figure 1.1: Distribution of training examples per sound event in the TUT Sound Events 2016/2017 and TUT Acoustic Scenes 2016/2017 datasets.

inactive frames and long duration events, potentially improving overall detection accuracy.

Imoto et al. approach the data imbalance problem in SED using various loss functions to control the training weights of sound events. They introduce a Focal Loss (FL) loss function to reweight loss contributions based on a sound event prediction confidence [6]. They then extend this work using loss functions such as Inverse Frequency Loss, Class-Balanced Loss, Asymmetric Focal Loss (AFL), and Focal Batch Tversky Loss [7]. AFL augments FL by applying a down-weighting factor to inactive training samples. Their work suggested that inactive frames are more detrimental to SED performance than inter-class imbalance, and achieved improved performance using AFL. Zhang et al. introduce a Unified Loss Function (ULF) which combines a class-sensitive loss term to AFL in order to simultaneously address inter/intra-class imbalance, and achieve state-of the-art performance [8]. A limitation of these approaches is their dependence on SED model prediction confidence to determine the relevant weighting factors on each training sample, but model confidence is often miscalibrated for under-represented classes. In addition, FL methods require fully labelled datasets for training, which is expensive to obtain, and are also highly sensitive to hyperparameter settings which requires extensive tuning for optimal performance.

Active Learning (AL) is a Machine Learning (ML) strategy that reduces the annotation effort needed to train models through the optimal selection of abundant unlabelled data. Given the cost of data collection and imbalance problem in SED, AL is a promising strategy to deal with both these issues concurrently. To this end, this dissertation introduces Spectral Entropy Active Learning (SEAL), which is an AL algorithm that undersamples redundant long duration sounds while prioritising selection of challenging short duration sounds to train SED models. It achieves this by exploiting the transient nature of short duration sounds, which typically display rapid spectral changes. By using the Spectral Entropy (SE) of the audio in the AL acquisition function, the approach avoids using model uncertainty which is unreliable in the early AL training stages. Accordingly, this research aims to answer the question:

"To what extent can Active Learning using Spectral Entropy as an uncertainty sampling selection metric, be used to reduce the effect of data imbalance in Sound Event Detection, as opposed to Asymmetric Focal Loss?"

1.2 Research Objectives and Scope

The objective of this work is to determine if AL can address the data imbalance problem. An AL algorithm aims to improve model performance by actively selecting the data from which it learns. By this definition, the study hypothesises that AL can achieve comparable performance to FL-based approaches while using fewer data. AL has been applied previously to SED tasks, but these did not specifically take data imbalance into account. This work introduces a novel sample selection strategy, SEAL, to mitigate data imbalance by prioritising selection of short duration sounds that are usually under-represented in the dataset. The study then evaluates SEAL's performance against AFL and other commonly used AL strategies. Concretely, the objectives of this research are to:

1. Determine if SE is an effective sample selection metric to detect active frames, and in particular short duration events.
2. Evaluate whether SEAL can achieve accuracy comparable to AFL, while requiring fewer labelled data to do so.

Satisfying these two objectives allows the study to adequately answer the research

question. In order to evaluate SEAL performance, experiments are set up using existing SED models and datasets from literature. Given the vast modalities of SED tasks, the scope of this research is limited to the following. Firstly, Deep Neural Network (DNN)-based SED models are used, as they are most widely adopted in state-of-the-art SED tasks. Secondly, only strong labelling is considered for annotating the audio training samples. This means the *exact* onset and offset time for each sound event is provided in the annotation, as opposed to weak labelling, which only indicates the presence of a certain event.

1.3 Research Contribution

The contributions of this work are as follows:

- It shows SEAL can improve class balance by prioritising selection of short duration sounds (Section 4.3.3).
- Given sufficient labelling budget, SEAL improves both overall model performance and recall on certain short duration sounds compared to reference AL methods, with comparable performance to AFL. (Section 4.3.1 and 4.3.2).
- The findings suggest that for highly imbalanced SED datasets, diversity and representativeness outperforms uncertainty-based AL strategies. (Section 4.3.1)

1.4 Organisation of Dissertation

The rest of this dissertation is organised as follows:

Chapter 2 reviews related work on AL for SED, and approaches to handling data imbalance in SED.

Chapter 3 provides the technical background relating to SED and AL.

Chapter 4 describes the SEAL algorithm, evaluation metrics, and results of the experiments. This is followed by a detailed analysis of the results.

Chapter 5 concludes the thesis by summarising the outcomes of the research questions, and provides directions for future work.

Chapter 2

Literature Review

This chapter reviews the related work in approaches to data imbalance and Active Learning in Sound Event Detection. Researchers have attempted to address the data imbalance problem by either data augmentation, model adaptation, or cost-sensitive loss functions. As for Active Learning, the trend points towards using hybrid acquisition functions over uncertainty or diversity selection on its own.

2.1 Approaches To Data Imbalance in Sound Event Detection

Data augmentation is the process of artificially generating new training samples to address inter-class imbalance. Mixup is a prominent augmentation strategy presented in [9]. It generates synthetic audio based on the linear interpolation of the raw waveforms and spectrogram embeddings of two reference audio samples. A tunable parameter controls the mixing ratio of the two input samples. Salamon et al. used various signal processing-based techniques, such as time stretching, pitch shifting, dynamic range compression, and adding background noise, to augment the UrbanSound8K dataset [10]. Park et al. present SpecAugment to augment automated speech recognition data [11]. Unlike time-domain-based transformations, SpecAugment directly modifies the audio spectrograms by randomly masking time segments and frequency bands, in order to improve the model's robustness to missing information during training. More recently, Generative Adversarial Network approaches have been adopted to synthesise artificial environmental audio data [12, 13].

The authors in [14] approach the data imbalance problem in SED by concurrently factoring the number of samples per event and their durations through a novel time-sensitive loss function. In doing so, they achieved improved performance on shorter events, such as *Dog* and *Dishes*. Nam et al. approach the problem at the model level, and introduce frequency dynamic convolution to address translational equivariance of 2D convolution in the frequency axis [15]. This resulted in improved performance on non-stationary sounds like *Alarm*, *Dog*, and *Cat*. Kim et al. combine feature pyramid networks and transformer encoders to better capture time correlations in sound signals to address the problem of varying durations across events [16].

Wang et al. use metric-based few-shot learning to identify similar-sounding events in a keyword detection in speech task. This approach only needs one or a few examples of the target class for learning [17]. Tonami et al. present a curriculum learning scheme for an acoustic scene classification task [18]. The approach guides the training of the SED model from easy to difficult-to-train events. The authors deem inactive samples as easier to model than active samples, and the approach achieves improved model accuracy over conventional training.

2.2 Active Learning for Sound Event Detection

Earlier AL schemes for SED systems were aimed at multi-class sound event classification, where only one of multiple labels to a candidate data sample need to be annotated. A clustering-based AL scheme is presented by the authors in [19]. In this approach, the unlabelled data pool is arranged into distinct clusters using K-Medoids (KMED) clustering, and the centroid of these clusters is annotated. Each sample within the cluster is then assigned the same label as its annotated centroid. A similar approach is presented using dictionary-based clustering in [20]. The clustering approach is extended with committee-based sample selection in [21]. It selects the candidate sample based on the mismatch of outputs of a nearest-neighbour and model-based classifier.

Wang et al. introduces Alternative Confidence sampling in a sound event classification task [22]. The method periodically samples high confidence candidate samples in order to correct previously incorrect annotations, and achieves improved results over reference methods. Shishkin et al. introduces Dropout Active Learning, which uses Monte-Carlo dropout to measure model uncertainty on candidate samples. The approach shows promising results on the UrbanSound8k dataset at low labelling budgets compared to KMED clustering [23].

Shuyang et al. present a comprehensive AL method which introduces several innovations [24]. Firstly, it implements change point detection to segment audio into clips containing complete sound events. Secondly, it employs weakly supervised learning, accommodating weak labels where audio clips contain known events but lack their temporal locations. Similarly to their previous work in [21], it uses *Mismatch First Farthest Traversal* as an acquisition function to select the most informative and diverse samples. This approach selects samples which highly disagree between different models (Mismatch), and then ensures the selection of samples farthest from those previously chosen, based on the cosine distance of their embeddings (Farthest Traversal). In contrast, the proposed approach relies on strongly labelled data, which is more laborious to obtain but provides exact temporal locations of events.

Martinsson et al. propose an adaptive change point detection method that aims to derive strong labels from weakly labelled data. The approach involves training a SED model on out-of-distribution data, which is then used to generate probability curves indicating the presence of target sounds. The sound event data containing the target sounds are segmented using change point detection and presented to the annotator. By focusing on segments containing target sounds, this approach can potentially mitigate the impact of inactive frames in imbalanced SED data [25].

Mohaimenuzzaman et al. present Deep Active Feature Learning (DAFL), which is AL approach that iteratively fine-tunes the audio feature extractor in each round of annotation [26]. This allows new feature embeddings to be used over time, which can be more discriminative than static embeddings used in conventional AL. In contrast to the spectrogram-based approach adopted in this study, DAFL processes raw audio waveforms as input features. Furthermore, DAFL employs the BADGE acquisition function to select samples based on gradient embeddings that encode both predictive uncertainty and distributional diversity, whereas the proposed approach uses SE as a proxy for uncertainty [27]. It is clear from literature that hybrid acquisition functions — combining both informativeness and diversity — is the preferred AL approach for SED.

Chapter 3

Background

This chapter explains the underlying techniques used in this work. It starts with a primer on Artificial Neural Networks, which underpin state-of-the-art Sound Event Detection models. It then details challenges in Sound Event Detection, such as data imbalance, and explores strategies to mitigate them. Finally, the chapter explains various Active Learning techniques and Spectral Entropy.

3.1 Artificial Neural Networks

An Artificial Neural Network (ANN) is a type of computational model inspired by neural structures in biological systems. It was first developed by McCulloch and Pitts, who proposed the idea of having an artificial neuron mimic brain-like processes [28]. The neuron is the fundamental building block of an ANN, and its function is to receive an input signal and produce an output or action. By cascading multiple neurons into many layers, the network is collectively able to learn complex mappings between a given input signal and a desired output via a learning algorithm. Such networks with many layers are commonly referred to as a DNN, and have shown tremendous capabilities in tasks such as pattern detection, natural language processing, and SED. In the development of ANNs, Rosenblatt's perceptron showed that networks with weighted connections between layers were capable of learning simple linear associations between an input and output signal [29]. Another key development was the backpropagation algorithm by Rumelhart et al. which facilitated efficient optimisation of the network's weights [30].

3.1.1 Feedforward Neural Networks

Feedforward neural networks are ANNs characterised by the flow of information going in only one direction (from input to output) with no feedback connections between node outputs. The class of networks with feedback connections between node outputs are known as Recurrent Neural Networks (RNNs), and are detailed in Section 3.1.5. The end-goal is for the network to learn a mapping f between the input x and output y, such that $y = f(x, \theta)$, given some parameters θ. These parameter values are learnt by the network through training. Their general structure consists of:

1. *input layer* - in this layer each neuron, also referred to as a node, represents a feature of the input signal.
2. *hidden layer* - these are a variable number of intermediate layers between the input and output layers, and perform transformations on the input layer to produce the output.
3. *output layer* - this is the layer that represents the final output or prediction of the network.

Figure 3.1 illustrates how nodes in different layers are interconnected. Starting at the first hidden layer, every node in each layer is interconnected to all nodes from the previous layer. The strength of these connections is governed by parameters known as *weights*. Each node output is effectively a weighted sum of all inputs from the previous layer. More formally, the output of the i-th neuron $z^{(i)}$ in the network can be calculated using:

$$z^{(i)} = \sum_{j=1}^{n} w_j^{(i)} a_j + b^{(i)} \tag{3.1}$$

Here $w_j^{(i)}$ is the weight connecting the j-th node from the previous layer to i-th node in the current layer, a_j is the *activation* of the j-th node in the previous layer, and $b^{(i)}$ is a bias term for the i-th neuron. Activation is analogous to biological neurons 'firing', or becoming active, when acted upon by stimulation from its environment. Importantly, the stimulus has to exceed a certain threshold in order for the neuron to fire. For instance, pain receptors in the arm should immediately trigger on a needle prick, and remain inactive to the touch of a feather. Similarly, in ANNs the

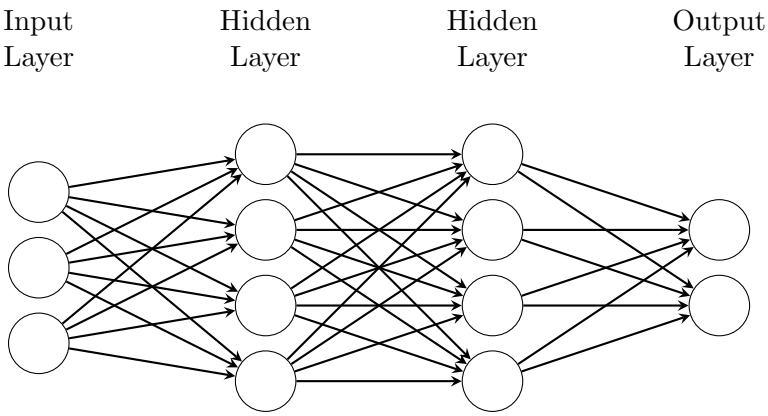


Figure 3.1: A simple fully-connected ANN with 3 input nodes, two hidden layers of 4 nodes each, and 2 output nodes.

activation is controlled by an activation function. These are differentiable functions and are necessary in order to add non-linearity into the network. Without activation functions, the network would simply be a linear model incapable of learning complex mappings. Additionally, these functions can also be used to bound the output values, which can assist with convergence and speed up training. Commonly used activation functions are listed below.

Sigmoid Function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

Used to map any input value to a range between 0 and 1. This is particularly useful in the output layer for a multilabel classification task i.e. the model predicts more than one class. Here, each node represents the probability of a particular class's occurrence, with the class assignment determined by an adjustable *threshold*. For example, if $\sigma(x) \geq threshold$, then the class is present. The threshold is typically set to 0.5, but can be adjusted to alter the class sensitivity.

Tanh (Hyperbolic Tangent) Function

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.3}$$

This function is typically used in hidden layers, and maps input values to a range between -1 and 1.

ReLU (Rectified Linear Unit) Function

$$\text{ReLU}(x) = \max(0, x) \tag{3.4}$$

The ReLU function outputs the input directly if it is positive. For negative inputs the output is 0. ReLU is mostly used in hidden layers and is simple to implement.

Softmax Function

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{3.5}$$

The softmax function is mostly used in the output layer of networks that perform multiclass classification i.e. the model predicts only one class from multiple options. It does this by squashing the raw output values into a discrete probability distribution.

3.1.2 Network Training

For supervised ANNs, the aim is for the network to produce a desired target output given an input. This is done through training, which is the process of iteratively updating the network's weights given a collection of input-output example pairs (x, y). The process starts with *forward propagation*, which passes x through the network to obtain an estimate of the output $\hat{y}$. The difference between this estimate and the target output y is measured using a *loss function*, and naturally the objective is to minimise this difference. Typical loss functions used in ANNs are mean squared error and cross-entropy loss, where the latter is calculated using:

$$\mathcal{L}(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{3.6}$$

Here $\hat{y}$ is the model's prediction and C is the number of classes for the classification task. Note y and $\hat{y}$ represent probabilities of a particular class occurring. The loss function is designed to heavily penalise model estimates which differ from the target value. After computing the loss, the process of backpropagation computes the gradient of the loss function with respect to each weight. These gradients indicate the direction and magnitude of change needed to minimise the loss. The network's weights and biases are then updated using an optimisation algorithm, which is explained in Section 3.1.3.

The collection of input-output example pairs is referred to as a dataset, and is divided into a training and a validation set. The reason for separate datasets is to gauge the model's generalisability — how well it performs on unseen data. The training set is used to update the model's θ parameters, while the validation set is used strictly to monitor the model's performance. Data leakage occurs when data from the validation set is inadvertently included in training, and should be avoided. The network is trained over several iterations called *epochs*, where each epoch consists of a full pass over the entire training set. For each epoch, the loss is monitored over the training and validation sets separately. If both the training and validation loss remain high throughout epochs, then the model is *underfitting*. This means the model is too simple to capture patterns in the input-output mappings. Conversely, *overfitting* occurs when the training loss decreases while validation loss is high, meaning the model is too complex and is learning example-specific patterns in the training data, leading to poor generalisability. To mitigate this, the training set typically contains much more examples than the validation set, in order to expose the model to a broader range of different input-output mappings, which improves generalisability. A common ratio between the training and validation set is 70:30 or 80:20 [31]. An effective model has a gradually decreasing training loss, and a validation loss which converges towards the training loss.

3.1.3 Optimisation

To minimise the loss function during training, various optimisation algorithms based on *gradient descent* are used. The loss gradient vector $\nabla\mathcal{L}$ with respect to the model parameters θ, is defined as:

$$\nabla\mathcal{L} \equiv \frac{\partial\mathcal{L}}{\partial\theta} \equiv (\frac{\partial\mathcal{L}}{\partial\theta_1}, \frac{\partial\mathcal{L}}{\partial\theta_2}, \ldots) \tag{3.7}$$

Starting from the output layer and moving towards the input layer, the gradient for each parameter is calculated using the chain rule of calculus. Each element in $\nabla\mathcal{L}$ represents the rate at which the loss $\mathcal{L}$ changes as the corresponding parameter θ varies. In turn, these gradients point to the direction of steepest ascent in loss. In order to reduce this loss, the parameters are updated in the opposite direction of the gradient. This update is performed using the following rule:

$$\theta \leftarrow \theta - \eta\nabla\mathcal{L}, \tag{3.8}$$

where η is a hyperparameter known as the *learning rate*. This process continues across epochs until the loss reaches a satisfactory minimum. This minimum is dependent on the differentiable loss function. The global minimum is the loss function's absolute lowest value, and local minima are the lowest values within different neighbourhoods. The choice of η affects the step size in each update: a large η results in more pronounced updates in θ, which can decrease training time, but may cause the loss to converge to a local minimum. Additionally, if the step size is too large, the loss value may oscillate about the minimum. A smaller η enables the loss to converge to the global minimum, but requires more updates to do so, increasing training time. In cases where η is too small, it may get stuck on a non-optimal local minimum.

The update rule in Equation 3.8 is referred to as batch gradient descent, which computes the gradient using the entire training set. This results in more stable estimates of the gradient, but becomes computationally expensive for larger datasets. A less computationally demanding alternative is stochastic gradient descent, where the gradient for a single randomly selected training example is computed in each iteration. This introduces noise from each individual example into gradient estimates, resulting in unstable updates. In some cases this allows the optimiser to escape local minima. However, excessive noise can prevent the optimiser from converging. Mini-batch gradient descent is an approach that combines the stability of batch gradient descent with the computational efficiency of stochastic gradient descent. It does this by using a small batch of samples from the training set. The size of the 'mini-batch' is usually a power of two (e.g. 16, 32, 64) in order to run optimally on specialised hardware, such as a Graphics Processing Unit (GPU).

Due to the large amount of training data available in modern settings, gradient descent is still prone to slow convergence and oscillations. *Momentum* is a technique that can be used to address this problem [32]. It includes a velocity term in the update rule that accumulates previous gradients, giving the gradient direction 'momentum'. The current update then becomes an exponentially decaying weighted average of past gradients. The velocity term for momentum is described as:

$$v_{t_e} = \Gamma v_{t_e-1} + \eta \nabla \mathcal{L}_{t_e} \tag{3.9}$$

and the update then becomes:

$$\theta_{t_e+1} = \theta_{t_e} - v_{t_e} \tag{3.10}$$

where Γ is the momentum coefficient that controls the influence of past gradients, and t_e is the epoch index. This approach allows for faster convergence by accelerating updates on stable slopes, while reducing oscillations with smoother updates for fluctuating gradients. It can also help the optimiser overcome shallow local minima, if the accumulated momentum is strong enough.

Adaptive Moment Estimation (ADAM) is a gradient descent algorithm that uses momentum and adaptive learning rates to improve convergence [33]. It uses the first moment of the gradient to accelerate updates for stable gradients, similarly to momentum. Additionally, it uses the second moment of gradients to adaptively scale the learning rate for each parameter based on their previous gradient magnitudes. The second moment is the exponentially decaying average of squared gradients. ADAM is described in Algorithm 1. The authors in the original paper used default values $\beta_1, \beta_2 = (0.9, 0.999)$ to control the decay rates for the first and second moments, respectively. ϵ is a small constant to prevent division by zero in the update rule. The initial moment estimates, which start at zero due to no previous gradients, are then bias-corrected by scaling them up early in training.

Algorithm 1: ADAM (Adaptive Moment Estimation)

Input: Learning rate η, Exponential decay rates $\beta_1, \beta_2 \in [0, 1)$, Small constant ϵ for numerical stability

Output: Optimised parameters θ

Initialise parameters θ_0

Initialise 1st moment vector $m_0 \leftarrow 0$

Initialise 2nd moment vector $v_0 \leftarrow 0$

Initialise timestep $t \leftarrow 0$

while θ_t *not converged* **do**

$t_e \leftarrow t_e + 1$

$g_{t_e} \leftarrow \nabla \mathcal{L}_{t_e-1}$ ▷ Compute gradient

$m_{t_e} \leftarrow \beta_1 m_{t_e-1} + (1-\beta_1) g_{t_e}$ ▷ Update biased 1st moment

$v_{t_e} \leftarrow \beta_2 v_{t_e-1} + (1-\beta_2) g_{t_e}^2$ ▷ Update biased 2nd moment

$\hat{m}_{t_e} \leftarrow m_{t_e} / (1-\beta_1^{t_e})$ ▷ Bias-correct 1st moment

$\hat{v}_{t_e} \leftarrow v_{t_e} / (1-\beta_2^{t_e})$ ▷ Bias-correct 2nd moment

$\theta_{t_e} \leftarrow \theta_{t_e-1} - \eta \frac{\hat{m}_{t_e}}{\sqrt{\hat{v}_{t_e}} + \epsilon}$ ▷ Update parameters

end

Regularisation is another key component to DNN optimisation, and primarily helps the network prevent overfitting. The regularisation techniques used in this work are listed below.

- **Dropout**: Srivastava et al. presented this algorithm as a mechanism to reduce overfitting in DNNs [34]. It randomly drops neurons from the input and hidden layers during training according to a predefined probability (usually between 0.1 - 0.3). As a result, these dropped neurons do not contribute to the output of the network, and forces the remaining neurons to learn the input-output mapping. This mitigates co-adaptation, which is the dependence of specific neurons between hidden layers contributing to the output, which in turn improves generalisability.

- **Early Stopping**: When training DNNs, the number of epochs is usually fixed, and it's assumed training cycles through all epochs. However with early stopping, training is prematurely halted if the validation loss does not decrease for a certain number of consecutive epochs. This number is a hyperparameter known as *early stopping patience*. Early stopping ensures the resulting trained network is not overfitting, and also reduces training time.

- **Batch Normalisation**: In some cases, backpropagation can cause numerical instability if errors propagated back through the network are too small (*vanishing gradients*) or too large (*exploding gradients*). Batch normalisation mitigates this by standardising the input to each layer for every mini-batch to have a zero mean and unit variance [35].

3.1.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of DNN that emulates the visual processing abilities of the human brain, and therefore excel at detecting patterns in spatial data. A landmark development was Lecun's LeNet model, which achieved state-of-the-art performance on a handwritten digit classification task [36]. Krizhevsky developed AlexNet, which was the first CNN to use ReLu activation and dropout, and it won the ImageNet Large Scale Visual Recognition Challenge in 2012 [37]. AlexNet's innovations marked a resurgence in the use of CNNs, as training such large networks was made more tractable. At the heart of CNNs is the process of *convolution*, which allows the network to detect localised patterns in the input data. A typical CNN structure consists of a convolutional, pooling, and fully-connected layer. Elements of each of these layers are described below.

- **Filters**: The convolutional layer is comprised of *filters* (also known as kernels), and is a rectangular matrix applied across spatial positions of the input to extract specific features from it, like shapes, edges, or colours. A filter slides

across the entire input to produce a feature map, which represents a single distinct feature. These feature maps are also referred to as *channels*, and exhibit a 'weight-sharing' property in that they allow the CNN to learn different features from the input in parallel. This operation also allows CNNs to be translationally invariant — each feature is detectable, regardless of its orientation or position in the input. The kernel value is element-wise multiplied with the input values at each position, and after summation the result is the corresponding feature map value. Padding is the addition of redundant data on the perimeter of the input to preserve its edge information. Stride is the number of steps each kernel slide takes — a stride length greater than one reduces the dimensions of the feature map, but may skip important feature characteristics in the input. Lastly, the filter size represents the dimensions of the filter matrix, where a smaller filter size is able to capture more fine-grained features.

- **Pooling**: After each convolution layer, a pooling layer is used to reduce dimensionality of feature maps while still retaining important features. Max pooling and average pooling are two common approaches, where max pooling selects the largest value under the kernel, and average pooling takes the average value under the kernel. Figure 3.2 illustrates the pooling operation for a (2x2) filter applied to a (4x4) input, with a stride length of two. Max pooling is useful in applications where detection of prominent features is preferred.

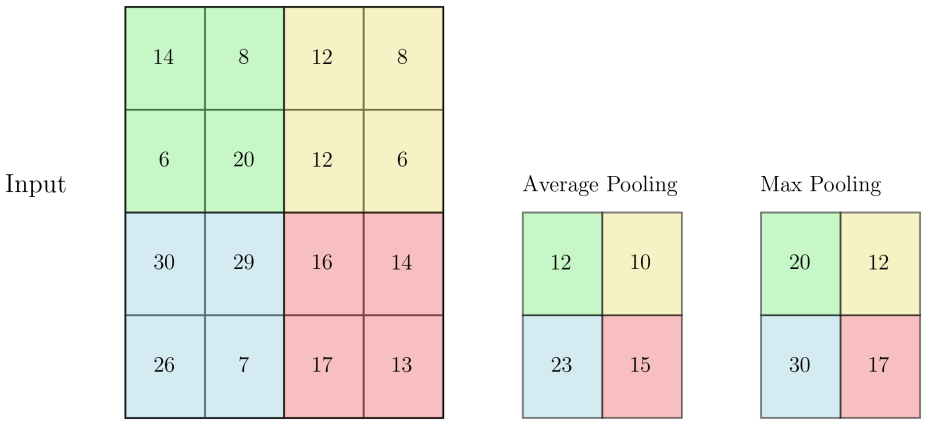


Figure 3.2: Illustration of Max and Average pooling operations, where corresponding coloured blocks represent the input and its pooled output.

- **Fully-connected layer**: This is the output layer of the network that connects to all activations in the previous layers. The input to this layer is flattened to

a one-dimensional vector. The layer is accompanied by a Softmax or Sigmoid activation function, in order to make output predictions based on the high-level features from the final pooling layer.

3.1.5 Recurrent Neural Networks

RNNs are a type of DNN designed to work with temporal data. While feedforward networks only use the current input x_t in each time step, RNNs have *hidden states* which preserve information from previous time steps, giving them a sort of 'memory'. As such, they are adept at working on tasks such as speech recognition and stock market price prediction [38, 39].

Given the input vector at time step t, and $\mathbf{x}_t = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T]$, the hidden state h_{RNNt} is calculated as:

$$h_{RNNt} = \sigma\Big(W\,\mathbf{x}_t + U\,h_{RNNt-1} + b\Big) \tag{3.11}$$

Here h_{RNNt} is the hidden state from the previous time step, W and U are RNN weight matrices, b is a bias term, and σ is an activation function. RNNs are still prone to vanishing and exploding gradients, and have since been improved upon. Hochreiter et al. introduced Long Short-Term Memory (LSTM) networks to address these problems, by using a gating mechanism which keeps track of information that is relevant and what was forgotten from previous states [40]. Its *cell state* (c_t), is comprised of three gates:

1. Forget Gate: Controls which historical information should be discarded.
2. Input Gate: Decides which current information should be added to the cell state.
3. Output Gate: Controls which information from the cell state should be used to compute the current output.

Bidirectional Recurrent Neural Network (BiRNN) are adaptations of RNNs that operate on temporal sequences in both the forward and backward directions [41]. It uses two RNNs to simultaneously process sequences from right to left, and vice-versa. This gives BiRNNs added contextual information, and is important in applications

like natural Language processing where the data has an inherent structure, as sentences tend to follow set patterns. A drawback of BiRNNs is that they need access to the entire input sequence at prediction time, which is not tractable for real-time applications.

Gated Recurrent Units (GRUs) are a computationally efficient alternative to LSTMs, which are also able to mitigate the vanishing gradient problem [42]. They introduce two new gates to control the flow of information: the reset gate (r_t) and update gate (u_t). The output at time step t, h_{GRU_t}, is a blend of the previous hidden state $h_{GRU_{t-1}}$ and the candidate hidden state $\hat{h_{GRUt}}$:

$$h_{GRU_t} = u_t \cdot h_{GRU_{t-1}} + (1 - u_t) \cdot \hat{h_{GRU_t}} \tag{3.12}$$

The candidate hidden state $\hat{h_{GRU_t}}$ depends on the previous hidden state $h_{GRU_{t-1}}$, the input x_t, and the reset gate r_t. If $r_t = 0$, the influence of the previous hidden state is ignored, which allows the GRU to reset and update based on the current input. When the GRU output is fed to a feedforward layer, contributions from the previous and candidate hidden states are:

$$c_{t-1} = w \odot (u_t \cdot h_{GRU_{t-1}}) \tag{3.13}$$

$$\hat{c}_t = w \odot \left((1 - u_t) \cdot \hat{h}_{\mathrm{GRU}_t} \right) \tag{3.14}$$

where w is a weight vector connecting the GRU to the feedforward layer, and $\odot$ is element-wise multiplication. The total output of the network is then:

$$o_t = c_{t-1} + \hat{c}_t \tag{3.15}$$

3.2 Sound Event Detection

The aim of a SED system is to temporally locate and classify distinct sound events present in an audio recording. More formally, it can be described as a multilabel classification problem where each time step in the recording can have multiple events

assigned to it. There are two processes involved in SED: audio representation and classification. In the representation stage, the audio data features are usually in the form of a time-frequency representation such as a Mel-Spectrogram. Panels (a) and (b) in Figure 3.3 shows the Mel Spectrogram for an office scene, with the corresponding sound event labels (event roll). A feature vector $x_t \in R^d$ is generated for each time frame t in the spectrogram. Here, $d \in \mathbb{N}$ denotes the number of features for each frame. In the classification stage, the SED model outputs the probability estimates $p(y_t(m)|x_t, \theta)$ of each sound class $m = 1, 2, ..., M$ in frame t, where θ is the model parameters obtained from supervised learning. Common models include CNNs, Convolutional Recurrent Neural Networks (CRNNs), and attention-based models such as the Audio Spectrogram Transformer [43–45].

3.2.1 Applications and Challenges

While SED is mostly confined to environmental sounds, there are various applications of audio processing and understanding that work with other types of target sounds, such as voice and music. For applications like automated speech recognition and music information retrieval, the objective is to extract qualitative information from the audio — such as phonemes in speech or tempo in music. In contrast, SED aims to quantitatively analyse audio data to detect and classify specific sound events. The IEEE Detection and Classification of Acoustic Scenes and Events (DCASE) research community is an entity focussed on the computational analysis of environmental sounds, and runs an annual competition on various sound analysis tasks, with many techniques developed for these tasks being directly applicable to SED [46]. These tasks include:

- Low-Complexity Acoustic Scene Classification: The aim is to identify the acoustic scene present in the audio recording, such as ‘metro station’ or ‘urban park’, given limited computational resources.
- First-Shot Unsupervised Anomalous Sound Detection: Detection of machine anomalies based on monitoring its audio during operation, with limited training data.
- Stereo Sound Event Localisation and Detection: Identify and find the position where sound events emanate from, in a video.
- Spatial Semantic Segmentation of Sound Scenes: Segment sound scenes spatially based on semantic meaning.

- Audio Question Answering: Answer questions based on audio content.
- Language-Based Audio Retrieval: Retrieve audio based on its natural language description. For example, “retrieve a sound clip containing a baby yelling as a woman talks”.

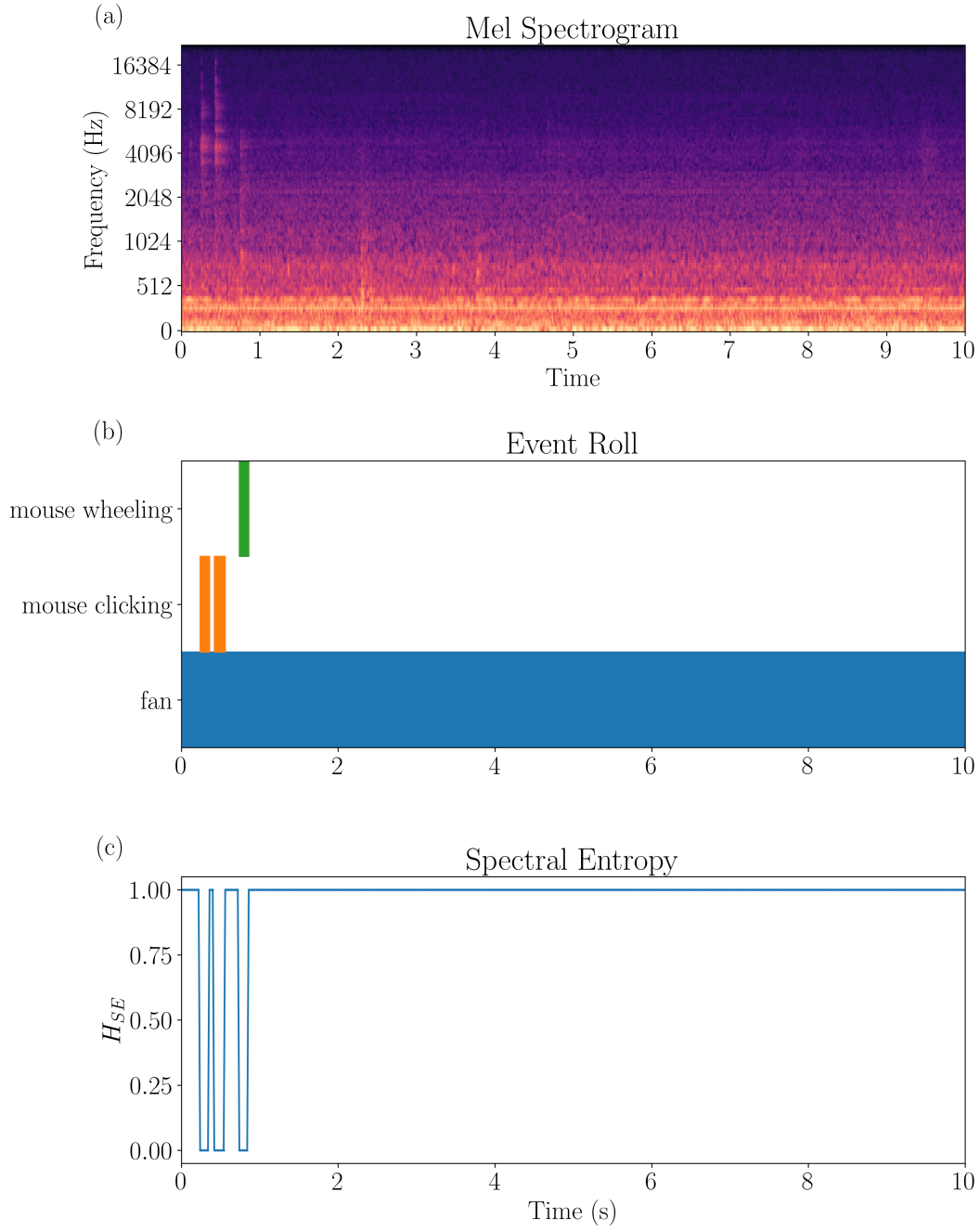


Figure 3.3: (a) Mel spectrogram of an audio signal, showing frequency content over time. (b) Event roll showing annotated sound events. (c) Instantaneous SE of the audio (MinMax scaled).

Apart from data imbalance, there are various challenges that hinder the performance and robustness of SED systems. These include sound occlusion, intra-class variability, and lack of labelled data.

Sound Occlusion

Occlusion is the masking of sounds by other sounds, or in other words, the overlapping of events. Researchers have attempted to address this problem by using sound separation techniques to help the model discern each event individually [47, 48]. Tonami et al. showed that by leveraging contextual information — like the acoustic scene of the audio — the model can better gauge the likelihoods of certain sounds occurring together [49]. In addition, microphone quality and placement dictate the amplitude levels of sounds. For instance, sounds located closest to the microphone will drown out faint sounds from afar.

Intra-class Variability

An event may not exhibit the same structure for all its examples in the dataset, due to differences in duration, recording environment, and sound source. For instance, a 'dog bark' is very different depending on the breed of dog. A robust SED system should aim to unify variations of examples for a particular event.

Lack of Labelled Data

Audio recording data are ubiquitous due to the low cost of microphone sensors. However, annotating this data is considerably more expensive. Also, the time needed for a human to annotate the data is at least equal to the duration of the data. In part, the work presented here aids in lowering the cost and time needed for data annotation. Crowdsourcing is a method which leverages online platforms to delegate the annotation process to a large group of people. A disadvantage of crowdsourcing is it presents issues in annotation quality, as non-domain experts or non-attentive listeners are more prone to mislabel ambiguous or complex examples. Martín-Morató et al. attempted to derive strong labels from crowd-sourced data, with the method achieving better precision than recall [50].

3.2.2 Feature Representation

Audio recordings from microphones represent the sound amplitude levels in the time domain. An analogue-to-digital conversion process transforms the data into digital audio, where its sampling interval and amplitude levels are quantised. WAV is a common digital audio format, where the amplitude level is defined by a bit-depth, and the sampling interval is defined by a sampling rate. CD Audio format is a popular WAV standard that uses 16-bit depth and 44,100 Hz sampling rate.

This work focuses on the Mel spectrogram representation of the audio. A Mel spectrogram is a matrix showing how the energy of different frequency bands evolve over time. A Mel filterbank, which is a set of triangular filters based on the Mel scale, is applied to the audio signal's Short-Time Fourier Transform (STFT) to generate a Mel spectrogram. Each filterbank corresponds to a frequency bin. The Mel filterbank gives better resolution at lower frequencies where human hearing is more sensitive. Taking the logarithm of the Mel spectrogram compresses the range of values in order to make quieter sounds more discernible. The steps to transform the digital waveform into a log-Mel spectrogram is summarised below.

1. **Windowing:** The digital waveform $x[n]$ is split into overlapping segments using a window function, such as the Hamming window $w[n]$. Each frame $x_f[n]$ can be represented as:
$$x_f[n] = x[n] \cdot w[n-t] \tag{3.16}$$
where t is the time index of the frame, and $w[n-t]$ is the window function.

2. **STFT:** The frequency spectrum for each frame $x_f[n]$ is obtained by computing its STFT:
$$X(t,f) = \sum_{n=-\infty}^{\infty} x_f[n] w[n-t] e^{-j2\pi f n} \tag{3.17}$$

3. **Mel Filter:** Values from the STFT are mapped into the Mel scale using Mel filterbanks, resulting in a Mel spectrogram. The Mel scale is defined as:
$$M(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{3.18}$$

4. **Logarithmic Mel Spectrogram:** The logarithm of the Mel spectrogram is taken in order to compress its dynamic range. The log-Mel spectrograms generated from each input are collated to give a final result that is a matrix with dimensions $d \times F$, where d is the number of Mel bins and F represents

the number of frames. The *frame width* is the length of the frame, and its *hop size* is the increment size of the window, and determines the level of overlap between frames.

3.2.3 Handling Data Imbalance

Conventionally, θ is trained using Binary Cross-Entropy (BCE) loss over all time frames T and sound event classes, which is defined as:

$$\mathcal{L}_{BCE}(\theta) = -\sum_{t,m=1}^{T,M} \{\mathbf{z}_{t,m} log(\mathbf{y}_{t,m}) + (1 - \mathbf{z}_{t,m}) log(1 - \mathbf{y}_{t,m})\} \tag{3.19}$$

where $z_{t,m}$ is the ground-truth label in time frame t and is 1 if sound event m is present, or 0 otherwise. $y_{t,m}$ is the model prediction of sound event m in time frame t. As BCE loss weighs the loss from each sample equally, losses from long duration and inactive frames will overwhelm training, and subsequently short duration and active sounds will tend to be ignored. AFL is one such approach to this problem, and is a common method in dealing with imbalanced data when training neural networks. It was originally applied to object detection in images, where background pixels tend to dominate over foreground pixels where the objects of interest lie [51]. This situation is analogous to SED, where frames can be dominated by background noise, silence, or monotonous sounds like fans blowing or murmurs from a crowd. AFL is calculated using:

$$\begin{aligned}\mathcal{L}_{AFL}(\theta) = -\sum_{t,m=1}^{T,M} \{&(1 - \mathbf{y}_{t,m})^{\gamma} \mathbf{z}_{t,m} log(\mathbf{y}_{t,m}) + \\ &(\mathbf{y}_{t,m})^{\zeta} (1 - \mathbf{z}_{t,m}) log(1 - \mathbf{y}_{t,m})\}\end{aligned} \tag{3.20}$$

where γ and ζ are parameters that control the weights of active and inactive frames, respectively. By setting $\gamma = 0$, the model focuses on down-weighting the loss contribution from inactive frames. Likewise, setting $\zeta = 0$ focuses on down-weighting the loss contribution from frames with high prediction confidence, which are deemed 'easy' frames. The assumption is that these frames correspond to inactive or long-duration sound events as they are easier to classify - and down-weighting them focuses learning on misclassified frames. The optimal combination of both these

components ($\gamma, \zeta > 0$) is shown to improve performance over using each component on its own [7]. Figure 3.4 shows the effect the γ parameter has on active frames — higher values makes the model less lenient towards incorrect (inactive) predictions. Although AFL shows improved performance over BCE in previous studies, it primarily addressed intra-class imbalance. ULF is an extension of AFL that incorporates a class-sensitive term on both active and inactive predictions to also mitigate inter-class imbalance. ULF is defined as:

$$\begin{aligned}\mathcal{L}_{ULF}(\theta) = -\sum_{t,m=1}^{T,M} \{w_+^m \mathbf{z}_{t,m} log(\mathbf{y}_{t,m}) + \\ w_-^m (1 - \mathbf{z}_{t,m}) log(1 - \mathbf{y}_{t,m})\}\end{aligned} \tag{3.21}$$

where the class-wise weights are computed as:

$$w_+^m = \alpha \left(\frac{C}{N_{m+} + \varepsilon} \right)^{\rho} (1 - y_{t,m})^{\gamma} \tag{3.22}$$

$$w_-^m = \beta \left(\frac{C}{N_{m-} + \varepsilon} \right)^{\tau} (y_{t,m})^{\xi} \tag{3.23}$$

As with AFL, the w_+^m and w_-^m terms in ULF include active/inactive weighting components γ, ζ, with the addition of active/inactive base weights α, β. C is the total number of frames in the training batch, with N_{m+} and N_{m-} the number of active and inactive frames for class m in the batch, respectively. The ϵ term is a positive constant to prevent division by zero, and ρ,τ are the respective inter-class balancing components. ULF can be converted to AFL by setting $\alpha = \beta = 1$ and $\rho = \tau = 0$.

3.3 Active Learning

AL is an iterative human-in-the-loop approach to ML. The objective of AL is to output a classification model given a set of initially unlabelled data and defined number of labels that can be manually labelled, known as the labelling budget. In pool-based AL, the process starts with a set of unlabelled data and an initial classification model. In each training cycle, the model selects a batch of unlabelled data based on a selection criteria to be sent to the human, also known as the oracle in AL literature, where its ground-truth labels are obtained. The model is then trained

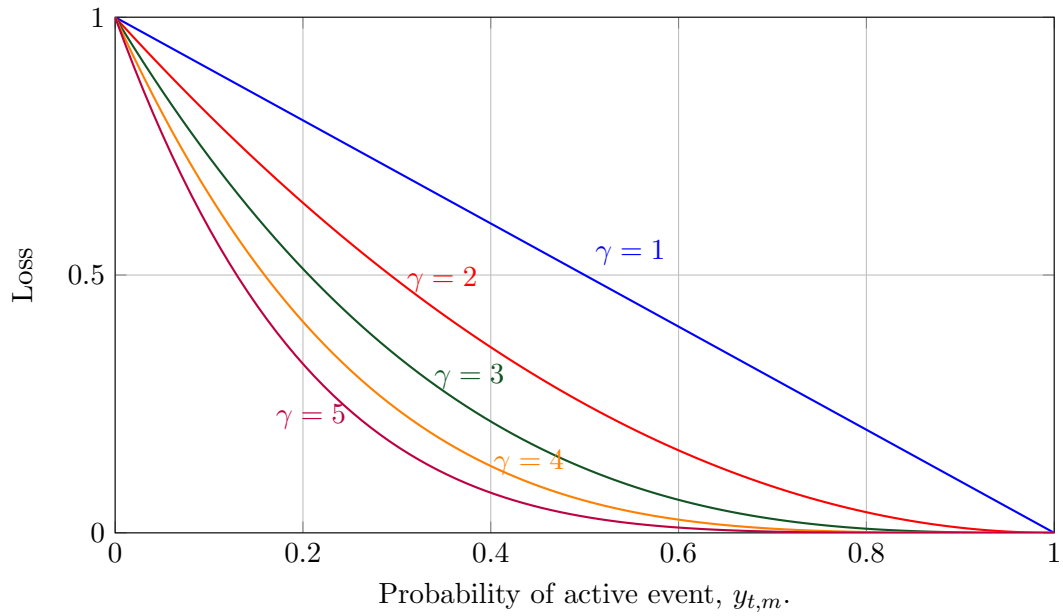


Figure 3.4: AFL loss for varying values of γ.

conventionally using the labelled samples. The process repeats until the labelling budget is exhausted, or a certain model performance level is reached. In order to optimise annotation effort, an AL algorithm aims to select the most informative samples using an acquisition function. This is in the belief that the learned model can achieve higher accuracy from less training data if the model chose what data to learn from. An optimal acquisition function enables the most informative samples to be selected, while also being diverse enough to be representative of the underlying data distribution. Common acquisition functions used in AL for SED are described below.

Least Confidence (LC)

This is an uncertainty-based strategy which selects samples with the lowest prediction confidence [22]. It is defined as:

$$x_{LC} = \underset{x_i \in U}{\operatorname{argmin}} \left(1 - P(y_{\max}|x_i)\right) \tag{3.24}$$

where x_{LC} is the sample selected, U is the set of unlabelled samples, and $P(y_{\max}|x_i)$ is the probability assigned by the current model to the predicted class with the highest probability for sample x_i. Other uncertainty-based strategies include using output probability entropy, and least-margin sampling, which selects samples with

the lowest margin between the top two class probabilities. To illustrate the intuition behind uncertainty-based sampling, Figure 3.5 shows features in a 2-D space for a toy binary classification problem. The gray points are unlabelled samples. The red circled region highlights samples that lie on or near the decision boundary, generated from a linear Support Vector Machine classifier. It's likely that these samples have high prediction uncertainty, and by labelling these points, the model can refine the decision boundary more effectively than if points on the extremities were labelled.

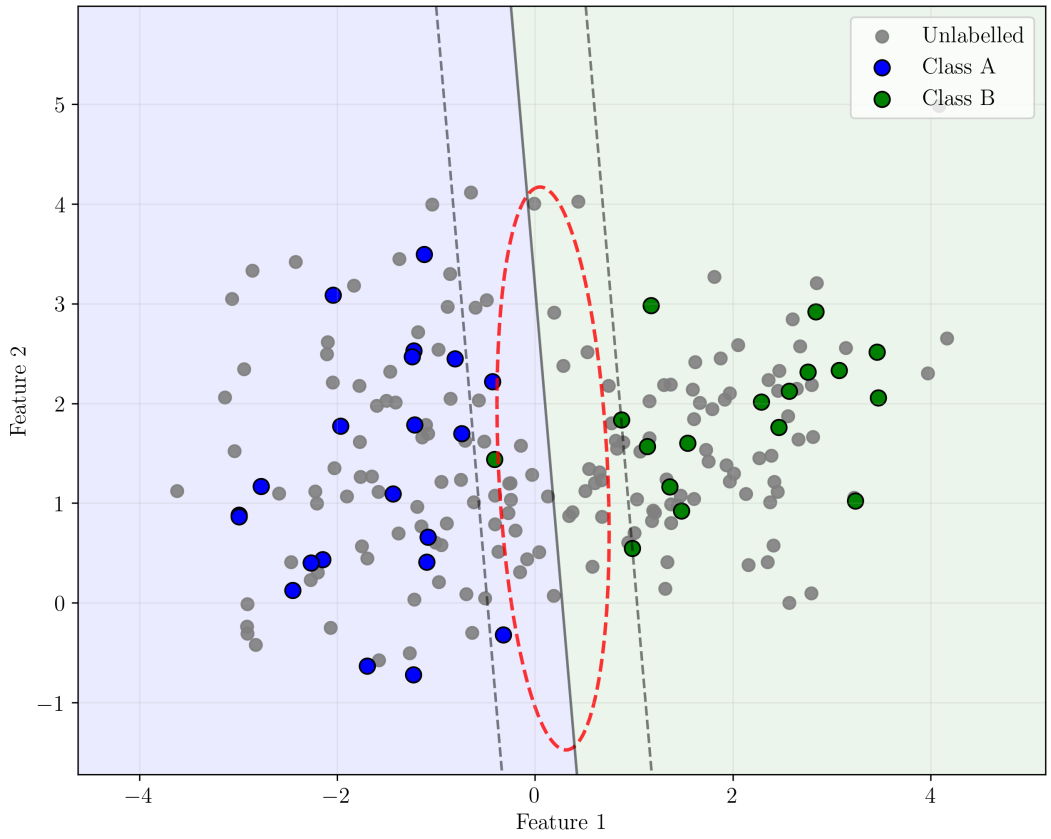


Figure 3.5: Toy binary classification problem showing a 2-D representation of features for the two classes, with its decision boundary. The red circle indicates samples with the most uncertainty.

KMED

Shuyang et al. use this approach to select samples most representative of the dataset [19]. It is a clustering algorithm similar to K-Means, but instead of using mean values of points within clusters to represent the cluster center, it uses actual data points as cluster centers. This makes it more robust to outliers than K-Means. These representative points are called *medoids*, and are the only points that need to

be labelled during each annotation round. The algorithm iteratively assigns data points to the nearest *medoid* to minimise the total dissimilarity within clusters, which is described as:

$$\arg\min_{S} \sum_{i=1}^{n} \sum_{j \in S} d(x_i, x_j) \tag{3.25}$$

where S is the set of indices of the chosen medoids, n is the total number of samples, and $d(x_i, x_j)$ is a distance measure (e.g. cosine distance) between samples x_i and x_j.

Random Sampling (RAND)

This is a commonly used baseline strategy which naively select samples at random. Various works showed that for imbalanced datasets, random sampling can outperform uncertainty-based sampling at low labelling budgets as it can lead to a more representative labelled set for training the model [52, 53].

3.4 Spectral Entropy

Given a Probability Mass Function (PMF), its information-theoretic entropy gives a measure of 'surprise' in the distribution. Similarly, the SE measures the level of disorder of a normalised spectrum [54]. The SE H_{SE} of a normalised spectrum is calculated as:

$$H_{SE} = -\sum_{i=1}^{N} z_i log(z_i) \tag{3.26}$$

where z_i is the normalised energy of the i-th frequency component of the spectrum and N is the number of points in the spectrum. To normalise the spectrum and convert it into a PMF, the following transformation is applied:

$$z_i = \frac{Z_i}{\sum_{i=1}^{N} Z_i} \tag{3.27}$$

where Z_i represents the energy of the i-th frequency component. Figure 3.3 (c) shows the SE computed for the audio signal of an office scene, and illustrates how SE drops

when short duration sounds with concentrated spectra occur, whereas SE increases for segments with noisy spectra. This property of SE allows us to distinguish short and long duration events based on their spectra.

Chapter 4

Spectral Entropy Active Learning Scheme for Sound Event Detection on Imbalanced Data

This chapter presents the proposed Spectral Entropy Active Learning scheme, along with details on the experiments to determine its effectiveness. It also presents and analyses the results from the experiments. The work presented is derived from the following journal paper:

> M.Y. Ally and L. Cheng, "Sound Event Detection on Imbalanced Data using Spectral Entropy Active Learning", to be submitted to the IEEE/ACM Transactions on Audio, Speech and Language Processing, 2025

The main research idea was conceived by M.Y. Ally, with guidance from Prof. L. Cheng. The software implementation of the SEAL algorithm and the experiments were coded by M.Y. Ally. The journal article was written by M.Y. Ally under the supervision of Prof. L. Cheng.

4.1 Spectral Entropy Active Learning

An AL scheme is proposed to address SED data imbalance by exploiting the properties of SE. Long duration sounds, similarly to noise, tend to have a flatter spectrum which results in high SE. Short duration sounds have well defined spectra concentrated in fewer frequency bands, and tend to have a lower SE. The aim is to build an acquisition function that prioritises short duration sounds, as it provides the following advantages:

i *Informativeness*: compared to long duration sounds, they possess more discriminative features owing to their transient nature and variable spectra.

ii *Robustness to model uncertainty*: Garin et al. showed that deep neural networks, which are commonly used as SED models, tend to be overconfident in their predictions, especially for under-represented classes or limited training data [55]. Therefore, using SE in the acquisition function instead of the model's output probabilities avoids relying on potentially overconfident uncertainty estimates.

iii *Robustness to amplitude variations*: SE enables the capture of spectral characteristics without being impacted by amplitude fluctuations of different events, as opposed to using Mel spectrogram features alone. For instance, a loud long duration event with high Mel energy could mask a co-occurring short duration event.

While SE is robust to amplitude variations, computing the SE over the entire duration of an audio clip may mask transient events when dominant long duration sounds are sustained. To account for this, the approach uses the *change* in SE over short segments of the clip. Given a log-mel spectrogram embedding $\mathbf{X} \in \mathbb{R}^{d \times T}$ of an audio clip, where d is the number of audio features (Mel bins) and T is the clip duration, the Spectral Entropy Change (SEC) is calculated as follows:

1. The audio clip $\mathbf{X}$ is split into $\Omega = \left\lfloor \frac{T}{S} \right\rfloor$ non-overlapping segments, each of duration S, producing sub-spectrograms $\{A_i \in \mathbb{R}^{d \times S}\}_{i=1}^{\Omega}$.

2. For each sub-spectrogram A_i, calculate its normalised energy distribution using:

$$P_i = \frac{\sum_{t=1}^{S} A_i(:, t)}{\sum A_i} \tag{4.1}$$

and its corresponding entropy:

$$H_i = -\sum_{k=1}^{d} P_{i,k} \log P_{i,k} \tag{4.2}$$

3. Calculate the absolute difference in entropy between consecutive segments, $\Delta H_i = |H_{i+1} - H_i|$ for $i = 1, 2, \ldots, \Omega - 1$, and sum these differences to obtain the SEC for the audio clip:

$$\text{SEC} = \sum_{i=1}^{\Omega-1} \Delta H_i \tag{4.3}$$

The SEAL procedure starts with a pool of unlabelled clips U, an empty labelled dataset D, labelling budget L, and an initial SED model θ. Additionally, K is a hyperparameter representing the batch size. The procedure runs for $\lceil \frac{|U|}{K} \rceil$ iterations, where each iteration selects a batch of K clips, and the total number of selected clips across all iterations must not exceed L. The selection procedure can be approached as a Maximum Coverage Problem: given the set U, the selected subset K at each iteration maximises informativeness and diversity [56]. The problem is known to be NP-hard; however, an optimal solution is approximated using a greedy strategy. Using SEC as the informativeness component of the acquisition function, diversity is enforced by selecting clips that are maximally distant from previously selected clips in the iteration. The set of selected clips $K^* = \{s_1, s_2, ..., s_K\}$ maximises the following objective function:

$$K^* = \arg \max_{\mathcal{C} \subseteq U, |\mathcal{C}|=K} \left(\sum_{i \in \mathcal{C}} \text{SEC}(s_i) + \sum_{i \in \mathcal{C}} \sum_{\substack{j \in \mathcal{C} \\ j \neq i}} d(s_i, s_j) \right) \tag{4.4}$$

Here $\mathcal{C}$ denotes a set of candidate clips in the batch. The distance measure $d(s_i, s_j)$ is the Cosine distance between two audio spectrogram embeddings, s_i and s_j, and is calculated using:

$$d(s_i, s_j) = 1 - \frac{s_i \cdot s_j}{\|s_i\| \|s_j\|} \tag{4.5}$$

where $s_i \cdot s_j$ denotes the dot product of the flattened embeddings s_i and s_j, and $\|s_i\|$ and $\|s_j\|$ represent the magnitudes of the flattened embeddings. The SEAL

procedure is summarised below.

Algorithm 2: SEAL

Data: labelled dataset D, unlabelled dataset U, labelling budget L, batch size K
Result: SED model
for $1 : \lceil \frac{|U|}{K} \rceil$ **do**
  if $L > 0$ **then**
    Train: θ updated using D;
    Sample selection: select K^* optimal clips;
    Annotate: Oracle assigns labels to K^* clips;
    Update:
      $D \leftarrow D + K^*$
      $U \leftarrow U - K^*$
      $L \leftarrow L - |K^*|$
  end
end

4.2 Experiment Setup

The experiment parameters are summarised in Table 4.1, and are derived from comparative SED and AL studies [7, 8, 57]. Experiments are conducted on Google Colab using Python version 3.11 and Pytorch 2.6.0. Additionally, a Nvidia T4 GPU is used to accelerate training computation. To evaluate SEAL performance, experiments are conducted using the TUT Sound Events 2016/2017 and TUT Acoustic Scenes 2016/2017 combined datasets [4,5]. It consists of 25 classes (see Figure 1.1) containing various everyday sounds from environments such as 'home', 'residential area', 'city centre', and 'office'. In total it contains 266 minutes of audio, of which 192 minutes for the development set and 74 minutes for the evaluation set, with manual annotations provided by [58]. 64-dimensional log-Mel spectrograms calculated for each 40 ms time frame with 50% overlap are used as acoustic features. Practically, it is not feasible for a human to discern a sound event in such short time frames, therefore the audio is segmented into longer clips with a 10 s duration. This makes it easier for the oracle to label each sound event and the start and end time of its occurrence, and allows for efficient use of annotation effort, since shorter clips would require labelling a larger number of separate clips. This clip duration is also convenient for benchmarking, as it is adopted in many DCASE task datasets [59].

Each input spectrogram has a corresponding ground-truth label based on the start

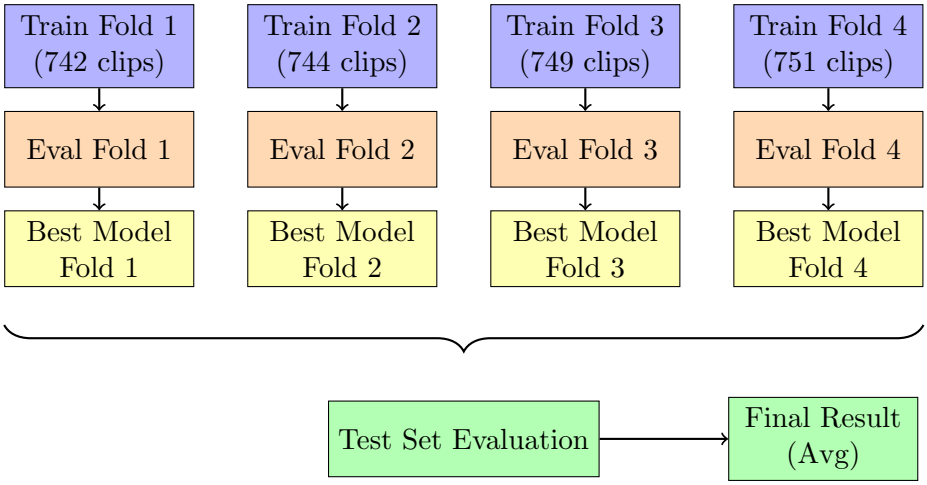


Figure 4.1: Illustration of cross-validation testing setup.

and end time in the annotation. The output of the model is a multi-hot vector of the sound events present in the input spectrogram. The SED model used is the CRNN-BiGRU, which is also used in DCASE 2023 Challenge Task 4B [57]. Model training uses a four-fold cross-validation setup on the development set and model parameters that achieve the best performance are selected. A cold-start approach is adopted in training each fold i.e. model weights are reset with random initial values for each fold at the start of the iteration in order to prevent data leakage. A *test set* is used to test the final model for each fold, and the average of the four fold models is taken as the result. The test set contains out-of-distribution data — these are clips which were not used for training or validation. Each experiment is run twice, giving a total of eight fold runs, where the average accuracy of the two experiments is reported. The cross-validation setup is illustrated in Figure 4.1.

SEAL is evaluated against the reference methods described in Section 3.3 by obtaining each method's model accuracy for varying labelling budget. BCE, AFL, and ULF are used as baseline methods, which are obtained using the entire labelled dataset. All of the AL methods are trained with BCE loss. The four folds in the development dataset contain 742, 744, 749, and 751 clips respectively, and the batch size is set to 32 clips to give a total of 15 iterations per fold. Increasing the batch size decreases the number of iterations and subsequent training time, but negates the effectiveness of the sample selection strategy, as larger batches are more likely to include redundant samples. Additionally, certain AL studies show that increasing the batch size typically leads to worse model accuracy [24, 60]. Segment-based ER and F1 score with micro and macro averaging are used as evaluation metrics, and the segment length is set to 1 s. These metrics are based on comparing the ground-truth labels with the model output over fixed segments. The metrics are detailed in Section 4.2.3.

Table 4.1: Experiment parameters.

Features	
Audio features (d)	64-dim. log-mel energy
Sample rate	44.1 KHz
Clip length (T)	10 s
Segment length (S)	1 s
Frame length / Overlap	40 ms / 20 ms
Batch size (K)	32
Model parameters	
Optimiser	ADAM
Learning rate	0.001
Dropout rate	0.3
Epochs	100
Early stopping patience	15
Detection threshold	0.5
CNN layers	3
Channels per CNN layer	128,128,128
Filter size	3x3
Max pooling kernel size	1x5,1x2,1x2
GRU hidden layers	32
Output layer size (M)	25
Loss function parameters	
AFL γ	0.0625
AFL ζ	1
ULF α	0.5
ULF β	1
ULF ρ	0.1
ULF τ	0.8
ULF γ	4
ULF ξ	4
ULF ϵ	1e-5

4.2.1 Data

The data is structured as a collection of WAV files, partitioned by the SED dataset and acoustic scene. The `dcase_util` Python package is used to perform various data manipulation tasks, such as loading SED datasets and generating log-Mel spectrograms from them [61]. Additionally, the `StandardScaler()` function from the `Sklearn` package is used to ensure consistency in the amplitudes of the audio data [62]. This function ensures all audio clips have spectrogram embeddings with zero mean and unit variance, across a fold, and is calculated using:

$$z_{scaled} = \frac{x - \mu_{fold}}{\sigma_{fold}} \tag{4.6}$$

where z_{scaled} is the transformed spectrogram based on the statistics of its fold.

For the cross-validation setup, each fold has a space-separated .txt file containing annotations for training and evaluation, respectively. Figure 4.2 shows an example of the annotation file. It contains the dataset file path, acoustic scene for the file, onset of event in seconds, offset of event in seconds, and the event label. Given that the input to the model is a (500×64) dimension log-Mel spectrogram, a corresponding target output for the same input clip is built using the annotations. The output is a binary matrix with dimensions (500×25), indicating 1 if an event is present in the particular frame, or 0 otherwise.

```
path scene_label event_start_time event_end_time event_label
audio/street/a127.wav city_center 0.0 65.024478 people walking
audio/street/a127.wav city_center 0.295063 51.291761 people talking
audio/street/a127.wav city_center 7.364277 20.150335 car
```

Figure 4.2: Example of annotations from `meta_fold01_development.txt`.

4.2.2 Model

While CNNs are adept at detecting patterns for simple events, they struggle to capture events with temporal dependencies or variable durations. Cakir's landmark CRNN model showed the utility of combining RNNs with CNNs to improve detection of these dynamic events [63].

The SED model presented here consists of three CNN layers, followed by a GRU layer. Its architecture is shown in Table 4.2, and the network layers are described in detail as follows.

Input (`Input`)

The input is a (500×64) log-Mel spectrogram. This represents a 10 s clip, where each of the 500 frames represents 20 ms (due to the hop size), with 64 Mel bins.

Convolutional layers (`Conv2d`)

There are three convolutional layers, each with 128 filters. The filter size is set to (3×3) with a stride length of one. This small filter size enables the convolutional layer to capture more fine-grained features.

Batch Normalization layer (`BatchNorm2d`)

Each of the convolutional layers is followed by a Batch Norm layer of the same dimension. Batch normalization standardises the output of the convolutional layer.

Max pooling layers (`MaxPool2d`)

Each batch normalization layer is followed by a max pooling layer, in order to reduce dimensionality of the CNN channels. In SED, frequency pooling is used, where only the frequency axis is pooled in order to preserve the temporal structure. This allows for better onset and offset detection of events.

Dropout layers (`Dropout`)

The output of each max pooling layer is fed to a dropout layer, which randomly drops 30% of activations during training, to help reduce overfitting.

GRU layer (`GRU`)

A GRU layer with 32 hidden units processes the reshaped sequence, in order to better capture temporal patterns in the clip.

Full-connected layers (`Linear`)

The GRU layer is followed by two fully-connected layers. This is then followed by a Sigmoid activation layer to get probabilities of each event occurring, for each time frame. An event roll can be generated from the Sigmoid activation output by binarising the probabilities if they exceed the threshold, as shown in Figure 4.3. In this work, the threshold is set to 0.5.

Table 4.2: SED model architecture.

Layer Type	Output Shape	Parameters
`Input`	(1, 64, 500)	0
`Conv2d(1, 128, 3x3)`	(128, 64, 500)	1,280
`BatchNorm2d(128)`	(128, 64, 500)	256
`MaxPool2d(1x5)`	(128, 64, 100)	0
`Dropout(p=0.3)`	(128, 64, 100)	0
`Conv2d(128, 128, 3x3)`	(128, 64, 100)	147,584
`BatchNorm2d(128)`	(128, 64, 100)	256
`MaxPool2d(1x2)`	(128, 64, 50)	0
`Dropout(p=0.3)`	(128, 64, 50)	0
`Conv2d(128, 128, 3x3)`	(128, 64, 50)	147,584
`BatchNorm2d(128)`	(128, 64, 50)	256
`MaxPool2d(1x2)`	(128, 64, 25)	0
`Dropout(p=0.3)`	(128, 64, 25)	0
`Reshape`	(500, 384)	0
`GRU(384, 32, bidirectional)`	(500, 64)	0
`Linear(64 → 32)`	(500, 32)	2,080
`Linear(32 → 25)`	(500, 25)	825
`Sigmoid`	(500, 25)	0

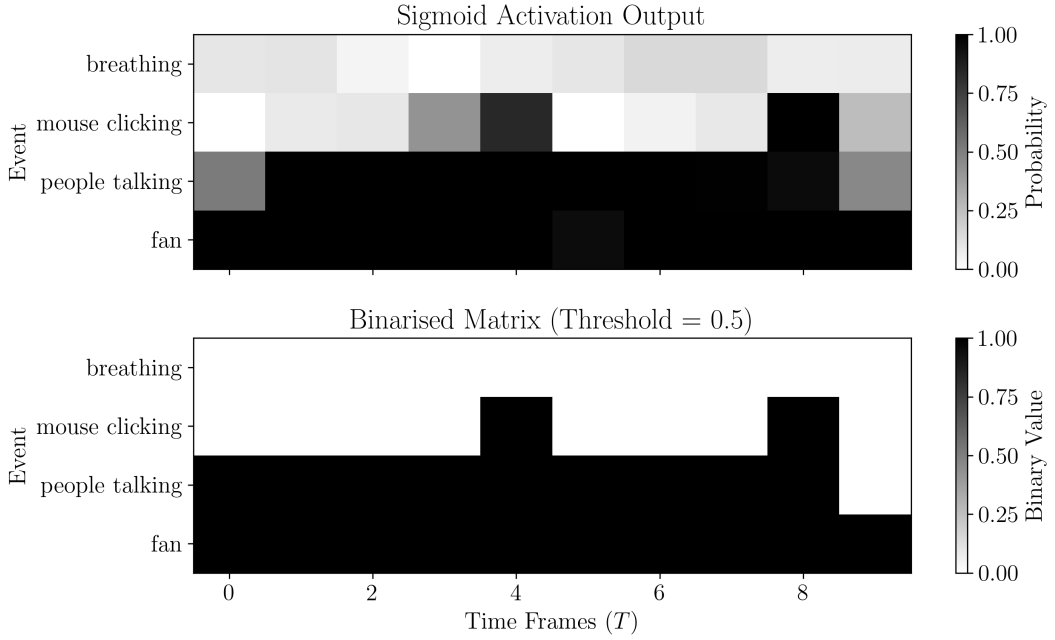


Figure 4.3: Binarisation of raw probabilities from Sigmoid activation output to generate an event roll.

4.2.3 Evaluation Metrics

F1-Micro Score

The F1-Micro score is a performance metric that considers the total true positives (TP), false positives (FP), and false negatives (FN) and true negatives (TN) across all events and time frames. TP and TN indicate whether an event is correctly identified as active or inactive in both ground truth and model output, respectively. FP means the model output incorrectly identifies an event as active, whereas FN means it missed an active event. F1-Micro score is calculated as:

$$\text{F1-Micro} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.7}$$

where

$$\text{Precision} = \frac{\sum_{i=1}^{M} \text{TP}_i}{\sum_{i=1}^{M} (\text{TP}_i + \text{FP}_i)} \tag{4.8}$$

and

$$\text{Recall} = \frac{\sum_{i=1}^{M} \text{TP}_i}{\sum_{i=1}^{M} (\text{TP}_i + \text{FN}_i)} \tag{4.9}$$

F1-Macro Score

This metric calculates statistics for each event individually. It treats all events equally by averaging their F1 scores, regardless of their distribution in the dataset. This makes it less sensitive to inter-class imbalance than F1-Micro score. The classwise F1 score (CF1) for a single event can be calculated using Equations (4.7)-(4.9), where the index i is set to the event of interest. The F1-Macro score is then calculated as:

$$\text{F1-Macro} = \frac{1}{M} \sum_{i=1}^{M} \text{CF1}_i \tag{4.10}$$

where M is the number of events.

Error Rate (ER)

A measure used to evaluate the performance of systems like speech recognition or text processing. It is based on three types of errors: insertions (I), deletions (D), and substitutions (S). Insertions and deletions correspond to FP and FN, respectively. Substitutions occur when the incorrect model output event is active compared to ground truth. Substitutions are counted as a single error, and not split separately as an insertion and deletion.

It is calculated as:

$$\mathrm{ER} = \frac{\sum_{k=1}^{T}(S_k + D_k + I_k)}{\sum_{k=1}^{T} N_k} \tag{4.11}$$

where:

- S_k is the number of substitutions for the k-th segment.
- D_k is the number of deletions for the k-th segment.
- I_k is the number of insertions for the k-th segment.
- N_k is the total number of events in the ground truth for the k-th segment.

4.2.4 Baseline Results

Table 4.3 summarises results from seminal works in data imbalanced SED. Both works use the same dataset, albeit with variations in their SED models. The CRNN-BiGRU shows significant improvement over the CNN-BiGRU for both F1-Micro/Macro scores. These results are used as a reference to gauge if the model and testing methodology are implemented correctly.

Table 4.3: Baseline results from seminal works in data imbalanced SED.

Baseline Results					
Loss	**Ref.**	**Dataset**	**Model**	**F1-Micro**	**F1-Macro**
BCE	[7]	TUT-SED16/17 TUT-ASC16/17	CNN-BiGRU	40.10%	7.39%
BCE	[8]	TUT-SED16/17 TUT-ASC16/17	CRNN-BiGRU	48.91%	20.03%
AFL	[7]	TUT-SED16/17 TUT-ASC16/17	CNN-BiGRU	48.29%	10.46%
AFL	[8]	TUT-SED16/17 TUT-ASC16/17	CRNN-BiGRU	52.07%	31.01%
ULF	[8]	TUT-SED16/17 TUT-ASC16/17	CRNN-BiGRU	54.39%	33.75%

4.3 Results and Analysis

This section presents the results from the experiments. The experiments are designed to gauge each method's respective overall (generalisation) and class-specific performance.

4.3.1 Overall Performance

Figure 4.4 shows the average F1-Micro performance across labelling budgets for all methods. Compared to the baseline BCE F1-Micro performance of 48.91% reported in [8], which uses the same CRNN-BiGRU model, the score of 46.19% achieved in this work falls within an acceptable range and supports the correctness of the model implementation. In contrast, the lower score of 40.10% reported in [7] is attributed to the use of fewer convolutional filters and training on weakly labelled data, which limits that model's ability to learn detailed patterns in the audio.

For low labelling budgets, KMED and RAND outperforms LC and SEAL, and approaches BCE and AFL performance. The high variance as indicated by the shaded regions suggest that the model predictions are more uncertain in the early AL stages. This explains the poor performance of LC, which is dependent on the model's prediction confidence. In contrast, RAND and KMED are model-independent strategies that select samples in accordance with the dataset's natural distribution and favour bringing in samples from the majority class. Since the test set is likely to

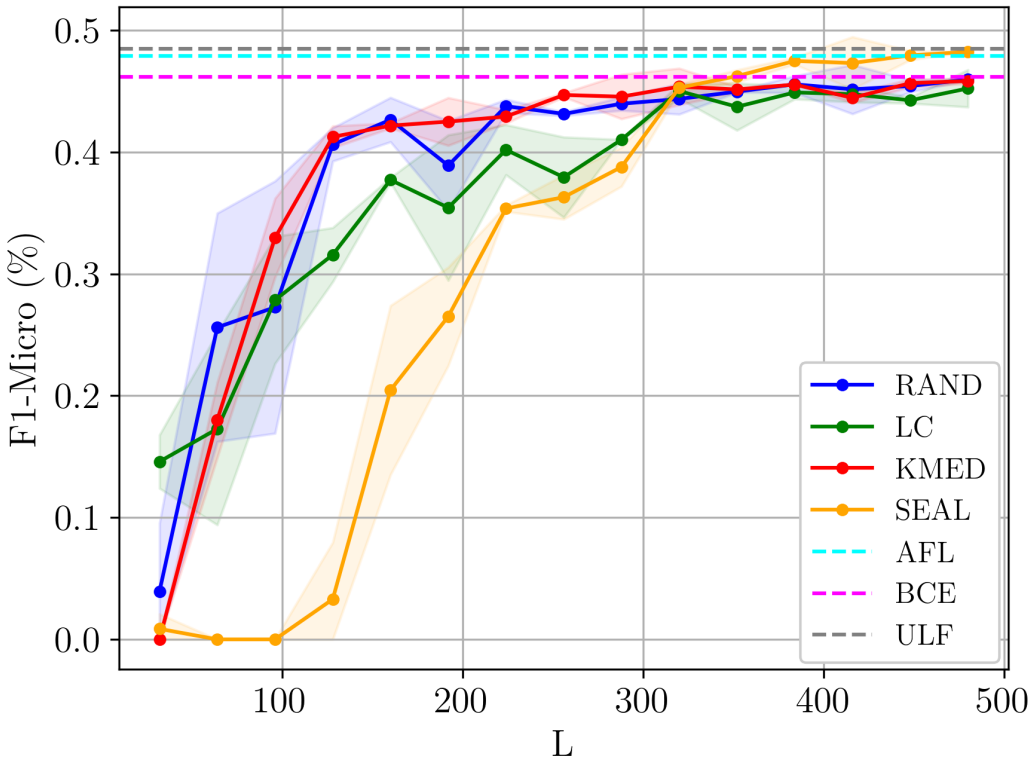


Figure 4.4: Average F1-Micro performance for varying labelling budget.

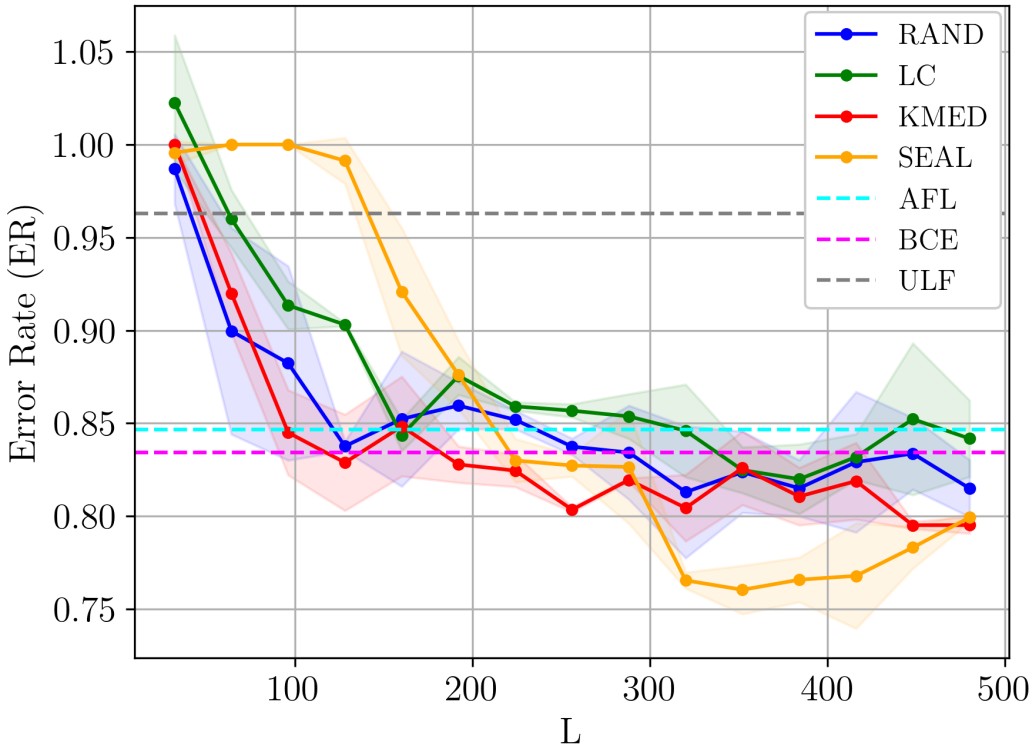


Figure 4.5: Average ER performance for varying labelling budget.

Table 4.4: Average error rates for AFL, ULF, and SEAL ($L = 480$).

	ER	Substitution Rate	Deletion Rate	Insertion Rate
AFL	0.847	0.185	0.341	0.321
ULF	0.963	0.225	**0.216**	0.522
SEAL	**0.799**	**0.153**	0.404	**0.243**

contain more examples from the majority class, RAND and KMED is able to achieve better performance from limited data. As the labelling budget increases, the model becomes more reliable and prediction variance decreases. SEAL achieves the best AL method performance after sufficient labelling budget of $L = 320$, which is about 40% of the entire dataset. It also surpasses AFL performance at $L = 480$, or about 65% of the dataset, with an average score of 48.23% compared to 47.92% for AFL. ULF shows the best performance with an average score of 48.47%.

At lower labelling budgets SEAL shows poor performance because it chooses samples which are under-represented in the dataset and whose acoustic features are most difficult for the model to discern. After the model has 'warmed up' by training on these challenging samples, SEAL performance is maximised. For this reason, SEAL can be considered a medium-budget AL strategy. The ER performance is shown in Figure 4.5. Similarly to the F1-Micro results, SEAL achieves the best performance at a sufficient labelling budget of $L = 320$. Although ULF maintains the best F1-Micro performance, it shows the worst ER performance. This can be attributed to its high insertion rate, as shown in Table 4.4.

4.3.2 Classwise Performance

Figure 4.6 shows the F1-Macro recall performance, which focuses on detectability of events. Similarly to F1-Micro performance, SEAL improves on reference AL methods after a sufficient labelling budget. At $L = 480$ it surpasses BCE but still falls below AFL and ULF performance. Again, KMED and RAND consistently outperform LC, but neither are able to surpass BCE performance. This suggests these strategies struggle to mitigate the impact of long duration events. Table 4.5 presents the average recall performance for each event at $L = 480$. It shows SEAL has a better recall on most events compared to other AL methods. It also surpasses AFL recall on events *Mouse Clicking*, *Keyboard Typing*, *Car*, *Cutlery*, and *(object) Rustling*. This shows the effectiveness of SEAL, as these are mostly short duration events that have rapidly changing spectra. Events *(object) Impact*, *Fan*, and *People Talking* have similar recall for both methods. SEAL shows lower recall on dominant long duration

events like *Large Vehicle* and *Bird Singing* compared to AFL.

Overall, ULF shows the best detectability, but generates more than twice as many false positives as SEAL, based on their respective insertion rates. This trade-off exemplifies how SEAL and ULF differ in approaching the SED imbalance problem. ULF's pronounced class reweighting is useful in applications like bioacoustic monitoring, where detectability of certain rare events is crucial. However, for applications where false positives are costly, such as medical sound monitoring, SEAL is better suited.

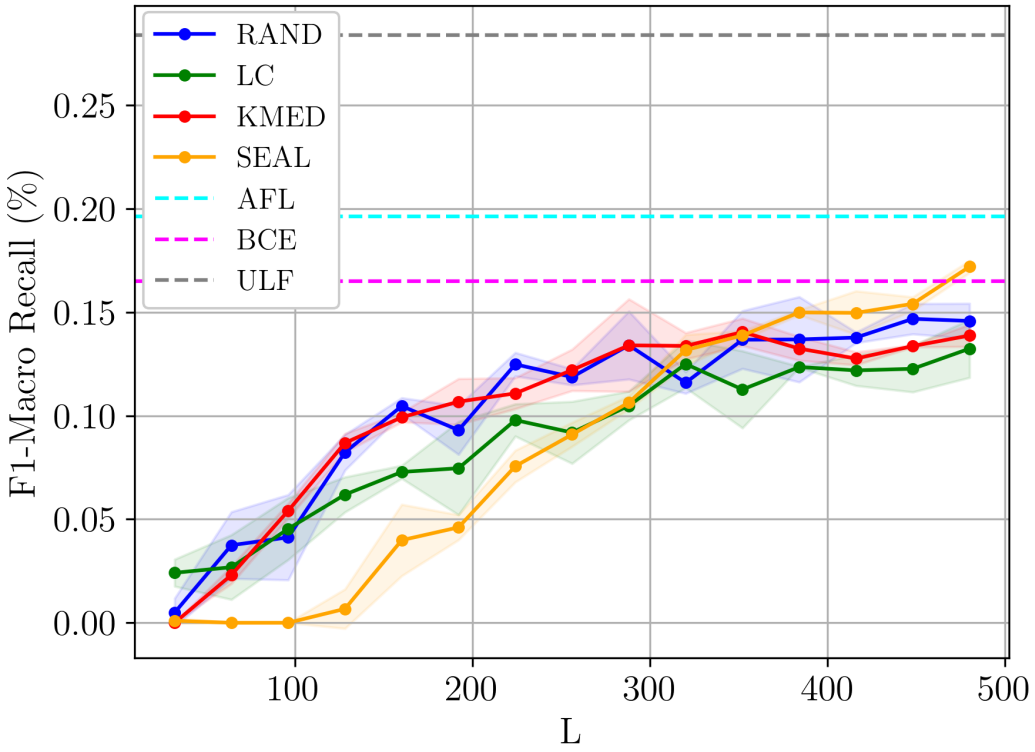


Figure 4.6: Average F1-Macro Recall performance for varying labelling budget.

4.3.3 Sampling Distribution

To better understand why SEAL is effective, an analysis is conducted on the samples selected at each labelling budget. Figure 4.7 shows the relative proportion of each sound event selected at a particular labelling budget for SEAL and RAND, respectively. RAND is chosen as the comparison baseline as it is the most consistent performing AL reference method. RAND favours the majority classes from the onset, with events such as *Fan* and *Car* taking a large proportion of the labelling budget. By contrast, at lower labelling budgets SEAL begins to select under-represented, short duration sounds like *Dishes*, *(object) Impact*, *Water Tap Running*, and *Cutlery*.

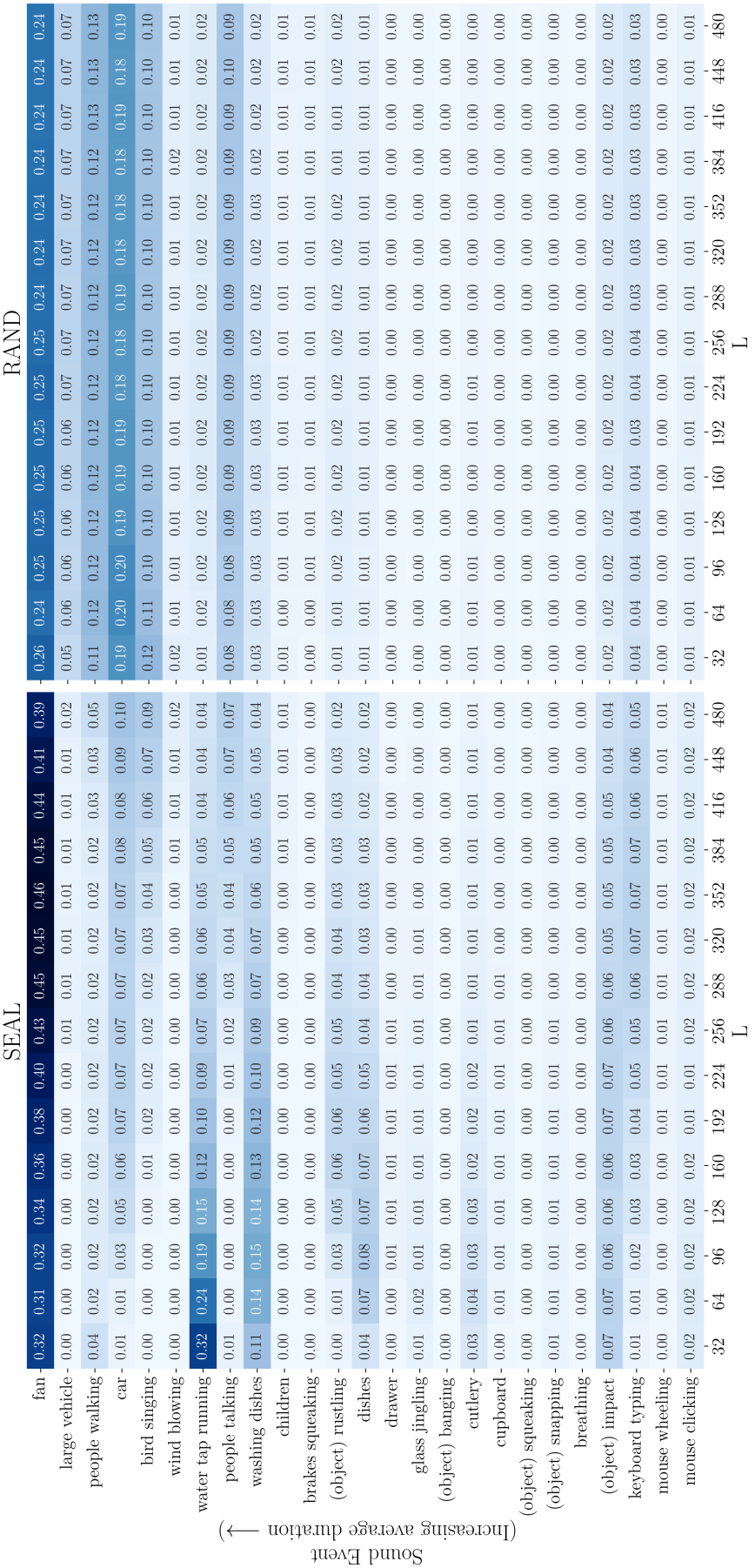


Figure 4.7: Sampling distribution for SEAL and RAND showing the proportion of labelling budget used by each event, for varying labelling budgets.

Even minority events like *Mouse Wheeling*, *Glass Jingling*, *Drawer*, and *Cupboard* are selected by SEAL, albeit in small proportions. These events would otherwise be ignored by RAND and shows that SEAL is able to better distribute labelling budget across events.

As labelling budget increases, SEAL still maintains a wider spread in sampling distribution compared to RAND. However, Figure 4.4 indicates this spread is not necessarily beneficial at low labelling budgets. This suggests that a strategy that prioritises representativeness at low budgets instead of selecting high SEC clips can improve low budget performance. As SEAL is a greedy strategy, the clips selected at each iteration tend to favour high SEC. Further to this, Table 4.5 shows events such as *Mouse Wheeling*, *(object) Snapping*, and *Drawer* are not even detected by SEAL, despite having increased sampling distribution. SEAL initially selected these events in higher proportions, but this dissipated as labelling budget increased. This suggests well-represented events dominated training in subsequent AL stages. A possible solution to this is to increase severely under-represented event samples using data augmentation techniques such as Mixup. Additionally, sample reweighting components from AFL or ULF can be combined with SEAL to better balance class weights at lower labelling budgets.

4.3.4 Analysis of Event Features

Certain groups of events tend to occur simultaneously and may have similar features, which lessens its detection accuracy. Figure 4.8 shows the audio features for a 20% subset of a random fold plotted in a 2-D space using Principal Component Analysis. Events such as *Washing Dishes* and *Water Tap Running* have overlapping features, which can make it difficult for the model to discern each event individually. Similarly, features for short duration events like *Mouse Clicking* and *Keyboard Typing* overlaps with *Fan*. This scenario is typical for audio recording scenes such as 'home' and 'office'. In this case, ULF up-weights the rare event activations, but inadvertently increases the transient event (*Fan*) activation as well. Consequently, ULF tends to misclassify overlapping events, as seen in its substitution rate of 0.225 compared to 0.153 for SEAL. This suggests SEAL is better equipped to mitigate the effects of sound occlusion.

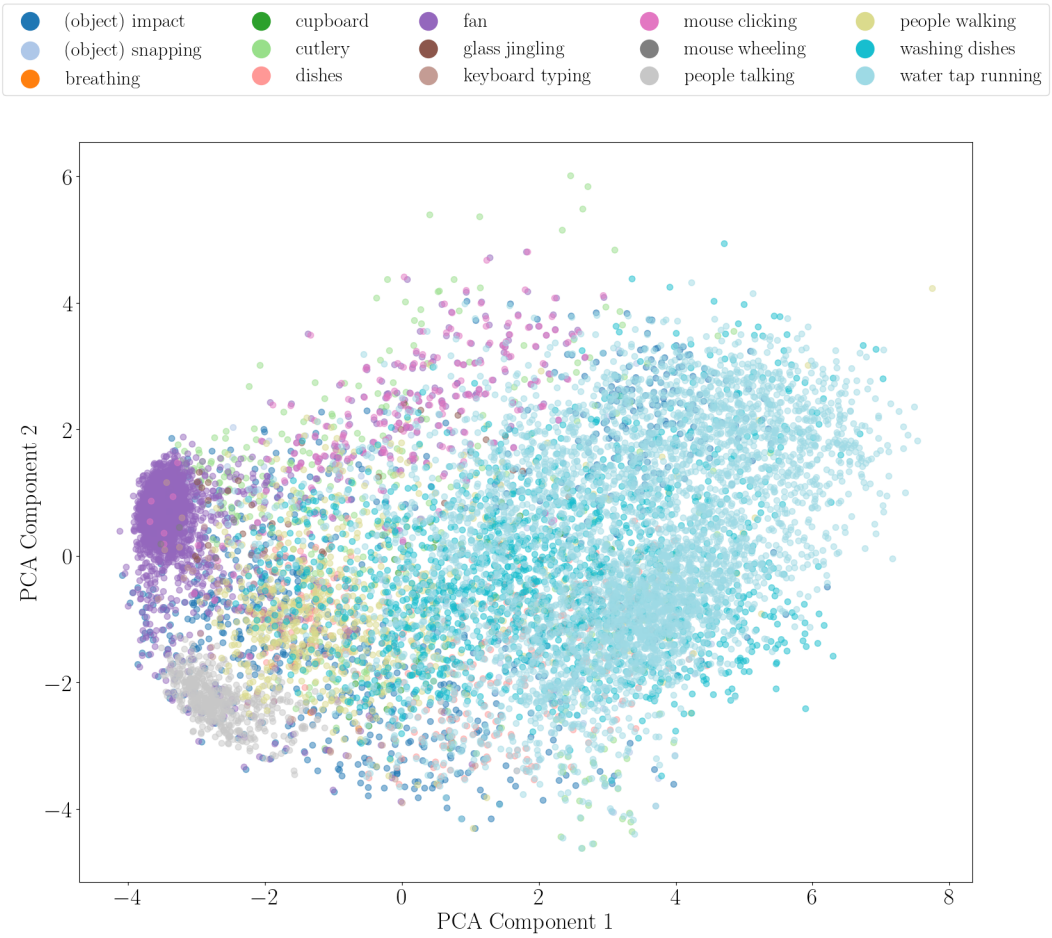


Figure 4.8: Plot of audio features in 2-D space using Principal Component Analysis.

4.3.5 Analysis of Loss Curves

Figure 4.9 shows the training and validation losses across epochs for SEAL and AFL, respectively. This experiment is confined to training on Fold 1, and selecting the optimal SEAL labelling budget of $L = 480$. All methods show acceptable loss values, and this is corroborated by their overall and classwise performances. SEAL maintains a stable margin between training and validation loss, which indicates the model is learning meaningful patterns in the data while avoiding overfitting. AFL also has a low margin between training and validation loss, but is slightly more erratic. This could indicate the class reweighting mechanism of AFL is down-weighting well-classified events, leading to poorer generalisation. Since the early stopping patience is set to 15, AFL prematurely stopped at epoch 36 because the validation loss had not improved since epoch 21. This early stopping results in less training time for AFL, which is an advantage over SEAL.

Table 4.5: Average classwise recall ($L = 480$).

Event	SEAL	KMED	LC	RAND	AFL	ULF
(object) banging	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
(object) impact	2.61%	0.30%	0.49%	1.39%	2.70%	**23.32%**
(object) rustling	8.93%	1.10%	0.82%	0.14%	3.98%	**28.57%**
(object) snapping	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
(object) squeaking	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
bird singing	41.03%	47.38%	52.76%	50.33%	54.86%	**59.85%**
brakes squeaking	0.00%	0.75%	0.00%	0.00%	1.60%	**31.52%**
breathing	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
car	82.70%	65.08%	70.83%	71.85%	75.86%	**82.87%**
children	0.00%	0.00%	0.00%	0.00%	0.00%	**5.56%**
cupboard	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
cutlery	0.74%	0.00%	0.00%	0.00%	0.00%	**10.66%**
dishes	1.02%	0.00%	0.71%	0.20%	1.83%	**10.47%**
drawer	0.00%	0.00%	0.00%	0.00%	0.00%	**0.23%**
fan	98.86%	99.46%	97.06%	99.25%	**99.50%**	98.94%
glass jingling	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
keyboard typing	59.78%	48.29%	9.27%	43.25%	56.05%	**75.00%**
large vehicle	3.15%	14.98%	17.23%	18.92%	24.89%	**35.59%**
mouse clicking	53.79%	16.52%	5.69%	21.54%	47.77%	**77.46%**
mouse wheeling	0.00%	0.00%	0.00%	0.00%	0.00%	**8.09%**
people talking	6.50%	1.93%	3.05%	5.89%	6.81%	**8.23%**
people walking	7.74%	11.33%	17.81%	11.11%	21.92%	**23.28%**
washing dishes	14.03%	19.11%	17.02%	15.81%	32.26%	**44.92%**
water tap running	35.71%	19.98%	24.87%	20.92%	44.09%	**56.21%**
wind blowing	13.48%	0.74%	13.48%	3.92%	16.42%	**28.92%**

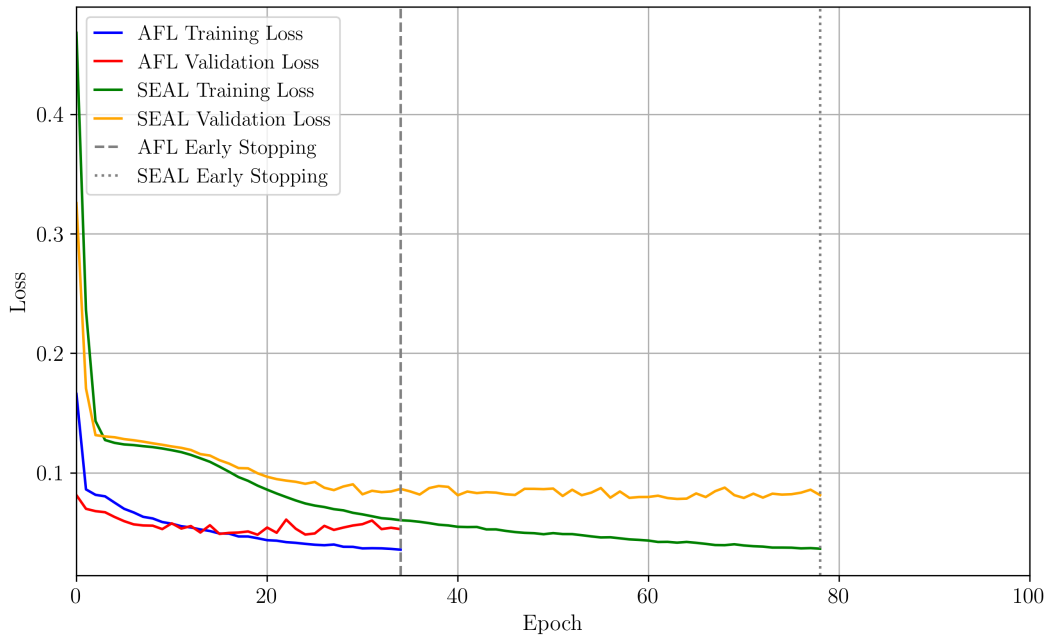


Figure 4.9: Training and validation loss across epochs for AFL (Fold 1) and SEAL (Fold 1, $L = 480$).

Chapter 5

Conclusion

5.1 Summary of Research

This work presents a novel SE-based AL scheme, SEAL, to address the data imbalance problem in SED. It first presents a literature review on related works in AL for SED, and addressing data imbalance. It also provides the technical background for ANNs, the reference AL methods, and SE.

It then details the SEAL algorithm, experiment setup, and analysis of results. Chapter 4 presents the results from the paper: "Sound Event Detection on Imbalanced Data using Spectral Entropy Active Learning", which is to be submitted to the IEEE/ACM Transactions on Audio, Speech and Language Processing, in 2025.

To reiterate, the contributions of this work are summarised below.

- SEAL improves labelling budget distribution across minority, short duration events such as *Dishes* and *Mouse Wheeling*, compared to RAND.
- SEAL achieves comparable overall AFL performance, with some improvement on classwise performance, given a sufficient labelling budget.
- A comparison of reference AL methods shows that diversity-based strategies (RAND, KMED) outperform uncertainty-based strategies (LC), especially at lower labelling budgets.

5.2 Recommendations and Future Work

For practical deployments of SEAL, a moderate labelling budget of 40%–65% of the entire dataset is recommended. At low labelling budgets, SEAL is outperformed by reference AL methods due to its prioritisation of hard-to-classify minority events, which lessens detection of majority events and degrades overall performance. This 'warm-up' requirement of SEAL means it needs the model to be trained on much larger amounts of minority event examples than what is available, in order to reliably detect these events.

There is an opportunity to mitigate the 'warm-up' requirement by *combining* SEAL with FL-based functions, like AFL or ULF. SEAL already shows promising performance with BCE at medium labelling budgets, but employing FL can up-weight majority events in the early AL stages, potentially leading to improved low budget performance, while still retaining SEAL's superior labelling budget distribution.

As discussed in Section 4.3.2, a tradeoff exists between detectability of events and false positive rates. As such, SEAL is recommended for applications where false alarms are costly, such as medical sound monitoring or industrial machine fault monitoring, due to its lower insertion rates compared to AFL and ULF, respectively.

5.3 Conclusion

To answer the research question, the two main research objectives are first validated:

1. Determine if SE is an effective sample selection metric to detect active frames, and in particular short duration events.

> The results from the sampling distribution analysis in Section 4.3.3 indicate the SE selection metric in SEAL does increase selection of short duration events, compared to the best performing reference method (RAND). The classwise recall performance in Section 4.3.2 further demonstrates SEAL's improved detectability of these events.

2. Evaluate whether SEAL can achieve comparable accuracy compared to AFL, while requiring fewer labelled data to do so.

> Section 4.3.1 shows that SEAL can achieve comparable AFL F1-Micro score, and surpasses it by 0.31 percentage points, with a 35% reduction in annotation effort.

Based on these findings, the research question can be answered:

"***To what extent can Active Learning using Spectral Entropy as an uncertainty sampling selection metric, be used to reduce the effect of data imbalance in Sound Event Detection, as opposed to Asymmetric Focal Loss***"

> This work demonstrates that using SE as an AL selection metric *does* reduce the effect of data imbalance in SED. Its prioritisation of under-represented, short duration events improves overall event sampling distribution, thereby mitigating the impact of inter-class imbalance. This manifests as increased recall on events such as *Mouse Clicking*, *Keyboard Typing*, *Cutlery*, and *(object) Rustling*, compared to AFL. However, the extent of this improvement is highly dependent on the labelling budget, with gains in model accuracy requiring a budget of at least 40%–65%.

References

[1] R. Alsina-Pagès, J. Navarro, F. Alías, and M. Hervás. homesound: Real-time audio event detection based on high performance computing for behaviour and surveillance remote monitoring. *Sensors (Basel, Switzerland)*, 17, 2017.

[2] J. Kotus, K. Lopatka, and A. Czyzewski. Detection and localization of selected acoustic events in acoustic field for smart surveillance applications. *Multimedia Tools Appl.*, 68(1):5–21, January 2014.

[3] F. Demir, A. Sengur, and V. Bajaj. Convolutional neural networks based efficient approach for classification of lung diseases. *Health Information Science and Systems*, 8, 12 2020.

[4] A. Mesaros, T. Heittola, and T. Virtanen. Tut database for acoustic scene classification and sound event detection. *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 1128–1132, 2016.

[5] 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 *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, pages 85–92, November 2017.

[6] D. Akiyama, K. Imoto, N. Tonami, Y. Okamoto, R. Yamanishi, T. Fukumori, and Y. Yamashita. Sound event detection using duration robust loss function. *arXiv preprint arXiv:2006.15253*, 2020.

[7] K. Imoto, S. Mishima, Y Arai, and R. Kondo. Impact of data imbalance caused by inactive frames and difference in sound duration on sound event detection performance. *Elsevier Applied Acoustics*, 196:108882, 2022.

[8] Y. Zhang, R. Togneri, and D. Huang. A unified loss function to tackle inter-class and intra-class data imbalance in sound event detection. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 996–1000, 2024.

[9] H. Zhang, M. Cisse, Y. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018.

[10] J. Salamon and J. Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

[11] D. Park, W. Chan, Y. Zhang, C. Chiu, B. Zoph, E. Cubuk, and Q. Le. Specaugment: A simple data augmentation method for automatic speech recognition. pages 2613–2617, 09 2019.

[12] A. Madhu and K. Suresh. Envgan: a gan-based augmentation to improve environmental sound classification. *Artif. Intell. Rev.*, 55(8):6301–6320, December 2022.

[13] C. Donahue, J. McAuley, and M. Puckette. Adversarial audio synthesis. In *ICLR*, 2019.

[14] S. Park and M. Elhilali. Time-balanced focal loss for audio event detection. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 311–315, 2022.

[15] H. Nam, S. Kim, B. Ko, and Y. Park. Frequency Dynamic Convolution: Frequency-Adaptive Pattern Recognition for Sound Event Detection. In *Proc. Interspeech 2022*, pages 2763–2767, 2022.

[16] S. Kim and Y. Chung. Multi-scale features for transformer model to improve the performance of sound event detection. *Applied Sciences*, 12(5), 2022.

[17] Y. Wang, J. Salamon, N. Bryan, and P. Juan. Few-shot sound event detection. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 81–85, 2020.

[18] N. Tonami, K. Imoto, Y. Okamoto, T. Fukumori, and Y. Yamashita. Sound event detection based on curriculum learning considering learning difficulty of events. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 875–879, 2021.

[19] Z. Shuyang, T. Heittola, and T. Virtanen. Active learning for sound event classification by clustering unlabeled data. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 751–755, 2017.

[20] W. Ji, R Wang, and J Ma. Dictionary-based active learning for sound event classification. *Multimedia Tools and Applications*, 78:3831 – 3842, 2018.

[21] Z. Shuyang, T. Heittola, and T. Virtanen. An active learning method using clustering and committee-based sample selection for sound event classification. *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, pages 116–120, 2018.

[22] Y. Wang, A. Mendez, M. Cartwright, and J. Bello. Active learning for efficient audio annotation and classification with a large amount of unlabeled data. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 880–884, 2019.

[23] S. Shishkin, D. Hollosi, S. Doclo, and S. Goetze. Active learning for sound event classification using monte-carlo dropout and pann embeddings. In *Proceedings of the 6th Workshop on Detection and Classication of Acoustic Scenes and Events (DCASE 2021)*, pages 150–154. DCASE, 2021.

[24] Z. Shuyang, T. Heittola, and T. Virtanen. Active learning for sound event detection. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2895–2905, 2020.

[25] J. Martinsson, O. Mogren, M. Sandsten, and T. Virtanen. From weak to strong sound event labels using adaptive change-point detection and active learning. In *2024 32nd European Signal Processing Conference (EUSIPCO)*, pages 902–906, 2024.

[26] M. Mohaimenuzzaman, C. Bergmeir, and B. Meyer. Deep active audio feature learning in resource-constrained environments. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:3224–3237, 2024.

[27] J. Ash, C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal. Deep batch active learning by diverse, uncertain gradient lower bounds. In *International Conference on Learning Representations*, 2020.

[28] W. Mcculloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Journal of Symbolic Logic*, 9(2):49–50, 1943.

[29] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65 6:386–408, 1958.

[30] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[31] A. Gholamy, V. Kreinovich, and O. Kosheleva. Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation. 2018.

[32] B. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[33] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[35] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[36] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[37] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[38] S. Islam, M.l Hasan, and M. Khan. Prediction of stock market using recurrent neural network. In *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0479–0483, 2021.

[39] R. Schmidt. Recurrent neural networks (rnns): A gentle introduction and overview. *ArXiv*, abs/1912.05911, 2019.

[40] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[41] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[42] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[43] H. Zhang, I. McLoughlin, and Y. Song. Robust sound event recognition using convolutional neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 559–563, 2015.

[44] R. Serizel, N. Turpault, H. Eghbal-Zadeh, and A. Shah. Large-scale weakly labeled semi-supervised sound event detection in domestic environments. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2018 Workshop (DCASE2018)*, pages 19–23, November 2018.

[45] Y. Gong, Y. Chung, and J. Glass. AST: Audio Spectrogram Transformer. In *Proc. Interspeech 2021*, pages 571–575, 2021.

[46] DCASE Community. Dcase challenge. `https://dcase.community`, 2025. (Online) Accessed: 26 March 2025.

[47] N. Turpault, S. Wisdom, H. Erdogan, J. Hershey, R. Serizel, F. Fonseca, P. Seetharaman, and J. Salamon. Improving sound event detection in domestic environments using sound separation. In *Workshop on Detection and Classification of Acoustic Scenes and Events*, 2020.

[48] T. Heittola, A. Mesaros, T. Virtanen, and A. Eronen. Sound event detection in multisource environments using source separation. In *First CHiME Workshop on Machine Listening in Multisource Environments (CHiME 2011)*, pages 36–40, 2011.

[49] N. Tonami, K. Imoto, R. Yamanishi, and Y. Yamashita. Joint analysis of sound events and acoustic scenes using multitask learning. *IEICE Transactions on Information and Systems*, E104.D(2):294–301, 2021.

[50] I. Martín-Morató, M. Harju, and A. Mesaros. Crowdsourcing strong labels for sound event detection. In *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 246–250. IEEE, 2021.

[51] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020.

[52] R. Pop and P. Fulop. Deep ensemble bayesian active learning : Addressing the mode collapse issue in monte carlo dropout via ensembles. *ArXiv*, abs/1811.03897, 2018.

[53] G. Hacohen, A. Dekel, and D. Weinshall. Active learning on a budget: Opposite strategies suit high and low budgets. In *Proceedings of the 39th International*

Conference on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, pages 8175–8195. PMLR, 17–23 Jul 2022.

[54] H. Misra, S. Ikbal, H. Bourlard, and H. Hermansky. Spectral entropy based feature for robust asr. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–193, 2004.

[55] Y. Gal, R. Islam, and Z. Ghahramani. Deep Bayesian Active Learning with Image Data. In *Proceedings of the 34th International Conference on Machine Learning (ICML-17)*, 2017.

[56] D. Hochbaum and A. Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics (NRL)*, 45(6):615–627, 1998.

[57] I. Martín-Morató, M. Harju, P. Ahokas, and A. Mesaros. Strong labeling of sound events using crowdsourced weak labels and annotator competence estimation. In *Proc. IEEE Int. Conf. Acoustic., Speech and Signal Process. (ICASSP)*, 2023.

[58] K. Imoto. Additional sound event labels of tut acoustic scenes 2016 & 2017. `https://www.ksuke.net/dataset/strong-sound-event-labels-of-tut-acoustic-scenes-2016-2017`, 2022. (Online) Accessed: 26 March 2025.

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

[60] G. Beatty, E. Kochis, and M. Bloodgood. Impact of batch size on stopping active learning for text classification. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 306–307, 2018.

[61] T. Heittola. Dcase util. `https://dcase-repo.github.io/dcase_util/index.html`, 2025. (Online) Accessed: 26 March 2025.

[62] F. Pedregosa et al. Standardscaler - scikit-learn documentation. `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html` . (Online) Accessed: 26 March 2025.

[63] E. Cakir, G. Parascandolo, T. Heittola, T. Virtanen, and H. Virtanen. Convolutional recurrent neural networks for polyphonic sound event detection. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 25(6):1291–1303, June 2017.