

Learning to rumble: automated elephant call and sub-call classification, detection and endpointing using deep architectures

Christiaan M. Geldenhuys  and Thomas R. Niesler 

Department of Electrical and Electronic Engineering, University of Stellenbosch, Stellenbosch, South Africa

ABSTRACT

We consider the problem of detecting, isolating and classifying elephant calls in continuously recorded audio. Such automatic call characterisation can assist conservation efforts and inform environmental management strategies. In contrast to previous work, in which call detection was performed for audio signals several seconds in length, we perform call activity detection at discrete time instants, which implicitly allows call endpointing. For experimentation, we employ two annotated datasets, one containing Asian and the other African elephant vocalisations. We evaluate several shallow and deep classifier models, and show that the current best performance can be improved by using an audio spectrogram transformer (AST). Furthermore, we show that transfer learning leads to improvements both in terms of computational complexity and performance. Finally, we consider automated sub-call classification using an accepted vocalisation taxonomy, a task which has not previously been considered, and for which the transformer architectures again provide the best performance. Our best classifiers achieve an average precision (AP) of 0.962 for binary call activity detection, and an area under the receiver operating characteristic (AUC) of 0.957 and 0.979 for call classification (5 classes) and sub-call classification (7 classes), respectively. These represent new benchmarks or improvements on previously best systems.

ARTICLE HISTORY

Received 2 December 2024
Accepted 25 March 2025


KEYWORDS


Elephant; automated call characterisation; passive acoustic monitoring; transformer; deep neural networks

1. Introduction

The African bush elephant (*Loxodonta africana*) and African forest elephant (*L. cyclotis*) have been identified as endangered and critically endangered species, respectively (Gobush, Edwards, Balfour, et al. 2020; Gobush, Edwards, Maisels, et al. 2020). The rapid decline in their population is primarily due to habitat loss and to poaching.

The African bush elephant is found predominantly in the open savanna, grassland and shrubland of sub-Saharan Africa, where they rely on vast territories to roam and forage for food. In contrast, the forest elephant inhabits the rainforests of Central and West Africa.

CONTACT Christiaan M. Geldenhuys  cmgeldenhuys@sun.ac.za

 Supplemental data for this article can be accessed online at <https://doi.org/10.1080/09524622.2025.2487099>

© 2025 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. The terms on which this article has been published allow the posting of the Accepted Manuscript in a repository by the author(s) or with their consent.

The Asian elephant (*Elephas maximus*) has also been identified as endangered by the IUCN. While this species is also hunted for its ivory, it faces the particular challenge of human-elephant conflict (Williams et al. 2020). The Asian elephant is found primarily in the dry forest and grassland of Southeast Asia, where it shares its habitat with rapidly growing human settlements. As these settlements expand, the natural habitat is increasingly fragmented and reduced. This has lead to increased competition for resources, with ensuing conflict between human and elephant.

The detection and classification of elephant vocalisations can provide important insights into the behaviour, distribution, and conservation status of these animals (Zeppelzauer and Stöger 2015; Keen et al. 2017). Automated behavioural classification can support passive monitoring and ecological management of wildlife in reserves and sanctuaries. This high-level information can be useful for bioacoustic and ecology research alike, as well as acting as an early warning system for threats, such as poaching. Furthermore, automated classification and endpointing are useful to downstream tasks such as passive sound source localisation (Geldenhuys 2023).

1.1. Contributions

In this study, we propose an automated system for the classification, detection and endpointing of elephant calls using deep learning architectures. We include a transformer-encoder architecture, which utilises a learnable embedding token to distinguish between different call types and has, to our knowledge, not been used to process elephant vocalisations before. Furthermore, we propose a novel transformer configuration that allows for improved endpointing performance. To develop our models, we make use of two corpora containing labelled recordings of both African and Asian elephant rumbles. We believe that this is the first work to use both out-of-domain and in-domain transfer learning to detect and classify elephant calls and the first to perform explicit call endpointing. Finally, we also perform explicit subcall classification, as a first step towards automated elephant behavioural classification.

2. Elephant vocalisations

African elephants have a complex vocal repertoire, consisting of various call types that serve different communicative functions and appear in different behavioural contexts. These calls can be divided into two main categories: laryngeal calls, which originate in the larynx, and trunk calls, which are produced by a blast of air through the trunk (Poole et al. 1988, 2005; Langbauer 2000; Poole 2011). These call types can be further distinguished based on their acoustic measurements, such as duration, minimum frequency, and spectral characteristics (Poole et al. 2005; Poole 2011). By examining how elephants use these calls in different behavioural contexts, we can gain insights into the complex dynamics of their social organisation and communication.

Laryngeal calls include rumbles, roars, grunts, cries, and idiosyncratic calls such as croaks and truck-like calls. Rumbles are low-frequency vocalisations that can travel long distances and are used in a variety of contexts, including greeting, reuniting, and reassurance. They can also convey information about the caller's identity, age, and reproductive status (Soltis 2010). Roars are loud, high-frequency calls that serve to

intimidate or warn other animals, while grunts are soft, low-frequency vocalisations used in close-range communication. Cries are high-pitched, urgent calls produced in response to separation or danger (Poole et al. 1988). The husky cry is a variant of the cry with a rougher, more strained quality. Croaks and *truck-like* calls are structurally unique calls that may not be socially relevant, with limited exemplars having been observed (Poole 2011).

Trunk calls include trumpets, bursts, snorts, and chirps. Trumpets are loud, high-frequency vocalisations produced by exhaling through the trunk, often used in greeting, excitement, or as a contact call (Poole and Granli 2011). Bursts are short, explosive calls produced by rapidly expelling air through the trunk, while snorts are similar but less intense. Chirps are high-frequency, bird-like calls produced by vibrating the tip of the trunk (Soltis 2010). The nasal trumpet is a variation of the trumpet call produced through the nose instead of the mouth, resulting in a softer, more muffled sound (Poole 2011).

3. Classification models

We will compare the performance of various classifier architectures when applied to the task of elephant call classification, detection and endpointing. In the following sections, we introduce these techniques.

3.1. Shallow classification models

In addition to current neural network classifiers, three shallow architectures were considered as baselines. These are described briefly in the following: logistic regression (LR), support vector machines (SVMs) and gradient boosting trees (specifically XGBoost).

3.1.1. Logistic regression

Logistic regression (LR) is a well-established linear approach to binary classification problems. The model estimates the log odds of an event's occurrence as a linear combination of the input features. The sigmoid function is then applied to the log odds in order to obtain a probabilistic output, used for classification. The posterior class probabilities are optimised through maximum likelihood estimation of the model parameters. Logistic regression (LR) is popular due to its simplicity, interpretability, and strong performance when the independent variables are linearly related to the log odds of the dependent variable.

3.1.2. Support vector machines

Support vector machines (SVMs) aim to find a hyperplane decision boundary that best separates data points of different classes (Cortes and Vapnik 1995). This hyperplane is typically chosen as the one with the largest margin, which is the distance between the hyperplane and the closest data points from each class, also known as the support vectors. SVMs use different kernel function transformations to transform data from a space in which they might not be linearly separable into a different space in which they are.

Commonly used kernel functions include linear, polynomial, radial basis function, and sigmoid kernels.

3.1.3. *Gradient boosting trees*

Decision trees recursively partition data into subsets based on feature values, leading to a tree-like structure. At each internal node of the tree, a test on an attribute is performed, and branches lead away based on the outcome of this binary test (i.e. true or false). The leaf nodes of the tree represent class labels in classification tasks. Decision trees can suffer from overfitting due to their tendency to create complex decision structures that memorise training samples rather than generalise patterns within the dataset (Quinlan 1986; Breiman 2001). To address this, various pruning strategies and tree ensemble methods have been proposed, such as reduced error pruning, cost complexity pruning, and random forests.

Random forests (Breiman 2001) are an ensemble learning method. A random forest consists of multiple decision trees trained on different sub-samples of the original dataset. Each of the trees uses a random subset of features to split nodes. The final classification is made by averaging the classification scores of all individual trees in the forest.

Gradient Boosting represent a popular ensemble technique that combines multiple weak performing models (e.g. decision trees) to build an overall model that preforms better than the individual weak component models. The primary goal is to improve the ensemble model estimation through sequential learning based on the weaknesses of previous models. XGBoost (XGB) (Chen and Guestrin 2016) uses a tree-based algorithm that focuses on minimising the objective function by iteratively fitting trees to the residual errors from the previous step.

3.2. *Deep neural classification models*

The shallow classifiers described above are well established and have already seen application in the field of elephant call detection (Clemins and Johnson 2003; Zeppelzauer and Stöger 2015; Silva 2017). Recently, neural architectures have come to represent the state-of-the-art in many machine learning applications. The following section describes three deep learning architectures, namely multi-layer perceptrons (MLPs), convolutional neural networks (CNNs) and transformer-based (encoder-only) and transformer-based (encoder-only) architectures. Furthermore, it describes the concepts of transfer learning and self-supervised learning through self-distillation.

3.2.1. *Multi-layer perceptron*

Multi-layer perceptrons (MLPs) (Rumelhart et al. 1986) are feedforward neural networks, consisting of multiple layers of neurons that process input data through a series of hidden layers to produce a final model estimate. Each neuron is fully connected to all neurons of the preceding layer and includes a non-linear activation function. Similar to LR models, MLPs are trained by maximising the posterior class probability.

3.2.2. Convolutional neural network

Convolutional neural networks (CNNs) (Fukushima 1980; LeCun et al. 1989) are a class of deep learning models traditionally applied in the field of computer vision and image processing. Originally inspired by the biological structure of the visual cortex, CNNs were designed to mimic human vision. The CNN architecture includes convolutional layers, pooling layers, and fully connected layers. Convolutional layers use small kernel filters to extract local features from the input, pooling layers downsample the feature maps to reduce dimensionality and computational cost, and fully connected layers enable the network to model complex decision boundaries. Over the last two decades, significant advances have been made in CNN architecture, such as AlexNet (Krizhevsky et al. 2012), VGGnet (Liu and Deng 2015), and residual neural networks (ResNets) (He et al. 2016). The latter employs skip connections between layers that allow deeper models to be more effectively trained.

3.2.3. Transfer learning

Transfer learning aims to improve model performance on a target domain by *transferring knowledge* contained in a different source domain (out-of-domain pre-training) or similar source domain (in-domain pre-training) (Pan and Yang 2009). Transfer learning has proven to be especially useful in low-resource or data sparse settings (Zhuang et al. 2020), as deep neural architectures typically require large quantities of data. Following pre-training, the model can be *fine-tuned* on the domain-specific data.

3.2.4. Transformers

The transformer architecture was first proposed by (Vaswani et al. 2017) as a successor to RNN for sequence-to-sequence natural language processing (NLP) tasks. Transformers have an encoder-decoder structure that relies on the attention mechanism to model both long- and short-term sequence dependencies. The architecture benefits from reduced training times, due to the consistent number of operations, compared to the recurrent networks that perform a variable number of operations, dependant on the input sequence length. As a result, the transformer architecture can be parallelised during training, while RNNs have to be trained sequentially. Furthermore, the global attention mechanism allows transformers to benefit from a constant dependence on path-length, which allows improved long-term and short-term dependency modelling. However, this comes at the cost of a memory requirement that is quadratic with sequence length.

3.2.5. Vision transformers

The standard transformer architecture, as proposed by Vaswani et al. (2017), was intended for machine translation and has since become the dominant method in NLP. A naïve application of the self-attention mechanism to images would require that each pixel attends to every other pixel. However, the quadratic memory requirement of the global attention mechanism, in terms of input sequence length, would make this practically infeasible for images. Thus, to apply transformers (encoder only) in the context of image processing, several modifications to the global attention mechanism have been suggested (Hu et al. 2018; Parmar et al. 2019; Zhao et al. 2020). Dosovitskiy et al. (2020) propose that a single two-dimensional image is divided into a sequence of smaller images, known as *patches*. A trainable linear projection is used

to reduce the dimensionality of each patch into a fixed size latent vector, known as a *patch embedding*. The process of obtaining a patch embedding is similar to the textual embedding process commonly employed in traditional NLP tasks. Similar to standard transformers, a positional embedding or encoding scheme (Vaswani et al. 2017) is applied to each patch embedding. This positional encoding remains one-dimensional, as two-dimensional encoding schemes have not proven more effective (Dosovitskiy et al. 2020). An optional learnable embedding token [CLS], can be appended to the start of the patch embedding sequence, the output of which serves as the model classification estimate (Devlin et al. 2019). The vision transformer (ViT) architecture allows image classification to make effective use of the self-attention mechanism, employed by the transformer-encoder architecture.

3.2.6. *Audio spectrogram transformer*

Gong et al. (2021) proposes the audio spectrogram transformer (AST) architecture, which performs audio classification by applying a ViT to a two-dimensional time-frequency spectral representation (mel-spectrogram) of an input audio sequence. Applying deep neural vision classification models to audio classification was first proposed by Hershey et al. (2017). Here, the authors applied a CNN architecture typically used for image classification to a two-dimensional representation of an input audio sequence.

The AST architecture first converts the input audio waveform into a 128-dimensional log-mel spectrogram representation, computed using a 25 ms hamming window with a stride of 10 ms (a typical configuration for speech recognition tasks). Each cell in this 2D input matrix can be understood to represent the confidence of a given frequency's presence within the signal at a particular time. The ViT architecture subdivides this matrix into several disjoint 'submatrices' (referred to as patches) each consisting of 16×16 cells and flattened into a single 256-dimensional vector. These patches are passed through a ViT model, with 768 embedding dimensions, 12 transformer encoding layers each with 12 self-attention heads (Gong et al. 2021). As in the ViT architecture, a learnable embedding token [CLS] is prepended to the start of the patch embedding sequence, the output of which is used to perform sequence classification.

3.2.7. *Self-supervised pre-training*

Annotating and labelling datasets is expensive because it typically requires human expertise. This makes techniques that do not depend on target labels attractive. *Self-supervised learning* is an approach where the target labels are either generated by the source data or using another algorithmic system. Self-supervised learning allows supervised training methods to be used on data previously only accessible to unsupervised techniques. Commonly, models are pre-trained using self-supervision, in order to obtain semantic knowledge of the underlying distribution, which may aid in further downstream tasks. These pre-trained models are then typically fine-tuned on a supervised dataset for which target labels are available. This fine-tuning dataset can be orders of magnitude smaller than the datasets used for pre-training. This allows models that were previously intractable to be employed. However, additional care should be taken to ensure the model does not overfit to the underlying distribution of data.

3.2.8. Bidirectional encoder representation from audio transformers

Chen et al. (2022) presents a ssl technique for iterative audio pre-training of bidirectional encoder architectures, such as transformer-encoders and bidirectional long short-term memory networks (BiLSTMs). The method consists of a teacher-student configuration in which semantic knowledge about the data is obtained, through iterative knowledge distillation, without the use of target labels. The method starts with a tokeniser in the form of a set of codebook embeddings that have been randomly initialised, and therefore contain no learned information. The input spectrogram is divided into 16×16 patches and passed through a learnable projection layer. The projected patches are compared using a nearest neighbour match with the codebook embeddings. The retrieved codebook embedding is used as the target label for the bidirectional encoder (teacher model).

The bidirectional encoder is then trained to map the given input spectrogram to the target labels, produced by the tokeniser. As for the tokeniser, the input spectrogram is divided into patches and passed through a linear projection layer. A random attention masking scheme is applied to the resulting patch embeddings, obscuring both features in time and frequency. Only the unmasked patch embedding is passed to the bidirectional encoder to obtain a latent representation of the respective unmasked patches. These unmasked latent representations, along with specific masked tokens in place of the masked latent representation [M], are passed to the label classification layer, typically a single linear layer with a sigmoid activation function. The predicted label, for each masked token, is compared with the target label generated by the tokeniser through a masked loss function, such as masked cross entropy. Hence, the loss is only evaluated between target labels and masked tokens. This in turn forces the model to embed semantic information about the predicted unmasked tokens within the masked tokens while only using a small subset of the input features. Once the bidirectional encoder has been trained, a new set of target labels is generated and used as ground truth values for the tokeniser model, discarding the previous tokeniser's target labels.

During subsequent iterations, the tokeniser codebook embeddings are trained on the new output labels from the encoder model. The new tokeniser target labels are compared to the teacher target labels through cosine similarity, and a set of learnable codebook embeddings are obtained. Once the tokeniser has been sufficiently trained, a new set of target labels is generated, which is used to retrain the encoder model.

This process of training the encoder model using the tokeniser labels and distilling the encoder class estimates into the tokeniser is repeated several times for a large unlabelled audio dataset, such as *AudioSet-2 M*¹ (Gemmeke et al. 2017) and is referred to as self-distillation (Chen et al. 2022). The result is a encoder model that has been pre-trained to estimate a set of quantised target labels from an unlabelled dataset. This pre-trained model can then be fine-tuned on a domain-specific source set, through transfer learning.

4. Literature review

In the following, we review the literature associated with two distinct objectives, namely *elephant call classification* and *elephant call detection*. Both fall within the realm of bioacoustic classification but have received limited attention in the research literature. Table 1 provides the quantitative summary of the results for the literature considered in this review.

Table 1. Summary of quantitative results for elephant call detection and classification reported in literature. Results are not directly comparable, as evaluations were performed on different datasets. *5-class*: 5 class call classification, *2-class*: 2 class call classification, *Det*: binary call detection.

Literature	Task	Accuracy	Precision	Recall/Sensitivity	Specificity
Clemins and Johnson (2003)	5-Class.	79.7	–	–	–
Venter and Hanekom (2010)	Det.	–	85.7	85.7	–
Stöger et al. (2012)	2-Class.	99.0	–	–	–
Zeppelzauer and Stöger (2015)	Det.	–	–	88.2	86.3
Keen et al. (2017)	Det.	–	–	87.2	91.1
Silva (2017)	Det.	–	84.0	84.0	–
Bjorck et al. (2019)	Det.	91.7	91.4	91.4	–
Leonid and Jayaparththy (2022)	Det.	96.2	93.2	88.2	87.2

Call detection is the task of determining whether a particular audio signal contains an elephant call or not. A related task in call detection is *call endpointing*, where the start and end of the call are identified within an audio signal. When call detection is performed at discrete time steps, the call endpointing is implicitly performed. To our knowledge, no work in the literature has focused on explicit call endpointing for elephant vocalisations. *Call classification* is a sequence classification task in which a short segment of audio is considered to identify the particular type of call. While call classification is always a multi-class task, call detection can be either binary or multi-class. To our knowledge, the literature currently does not consider explicit multi-class elephant call detection. Call classification can be viewed as a down stream task of either call detection or call endpointing.

Leong et al. (2003) worked on standardising the classification of African elephant calls, utilising measurements such as bandwidth, sound quality, fundamental frequency, infrasonic elements, and duration. The authors identified eight distinct call types, including three rumble variants that were distinguished by their bandwidth. A cross-correlation analysis was performed on the fundamental frequency contour of all rumble calls, the most common type, revealing five rumble categories. However, multidimensional scaling showed minimal clustering among call types, implying either overlapping rumble types or that the fundamental frequency contour may not be the primary physical property conveying the meaning of these signals.

The first application of speech processing techniques to the classification of elephant vocalisation signals, that we are aware of, was conducted by Clemins and Johnson (2003) and Clemins et al. (2005). These authors considered mel frequency cepstral coefficient (MFCC) features, manually adjusting the mel-scale to account for the infrasonic nature of elephant calls. Along with MFCCs, the log frame energies of the recordings were used to classify elephant vocalisations using a hmm. The authors obtained a 5-class (croak, rumble, rev, snort and trumpet) average call classification accuracy of 79.74 on a balanced set containing 74 vocalisations. The classification accuracy improved to 94.29 when omitting noisy recordings, using the same dataset and labels as Leong et al. (2003). Wood et al. (2005) went on to show that, by applying model-based cluster analysis to a set of acoustic features extracted from elephant rumbles, they could identify four distinct clusters when using frequency contour features and three clusters when using extracted acoustic parameters. Each of these clusters corresponded to specific observed animal behaviour, which corresponds with the findings of Leong et al. (2003).

Venter and Hanekom (2010) applied a subband pitch detector, originally developed for voice activity detection, to achieve elephant rumble detection by locating audio segments within which the pitch varied less than a predetermined threshold. The algorithm consists of two steps, first the pitch is estimated for an entire audio recording, and subsequently segments of low pitch variability are extracted. Each extracted segment is considered a detected elephant rumbles. The algorithm is only evaluated on a segment-level, with no results given on the boundary endpoint performance. This algorithm achieved for both segment-level precision and recall a score of 85.7. Stöger et al. (2012) applied (LDA), an SVM and a k-nearest neighbours (k-NN) classifier to distinguish between *oral*- and *nasal* rumbles. Each classifier used features obtained from a LPC smoothed spectrogram representation of the recordings, obtaining 99% Unlike previous approaches that relied on call-specific characteristics (such as predefined formant frequencies), this approach was fully automatic and made no a priori assumptions about the call structure.

The work described above was performed in either a controlled or a captive setting, which may not be representative of actual field conditions. Zeppelzauer et al. (2013) and subsequently Zeppelzauer et al. (2015) were the first to apply automatic classification techniques to free-roaming elephants. Two techniques were applied to elephant rumble detection, the first consisting of an SVM classifier using Greenwood cepstrum features with additional spectral signal preprocessing, and the second a template matching algorithm. In the case of the SVM classifier, a spectral representation of the entire audio signal is obtained, eight of the short time Fourier transform (STFT) frames are averaged before computing the Greenwood cepstrum features. A classification is made once for each of these aggregated frames (approximately every 120 ms). The template matching algorithm achieved a sensitivity of 78.6% and a false discovery rate of 21.4% and was outperformed by an SVM classifier, for which the corresponding figures were 88.2% and 86.3%. Keen et al. (2017) proposed a set of handcrafted two-dimensional convolutional kernels (Leung and Malik 2001) and, using an SVM, random forest (RF) and AdaBoost classification algorithms, improved on the state-of-the-art set by Zeppelzauer et al. (2015) for elephant rumble detection, achieving a sensitivity of 87.2 and specificity of 91.0. Silva (2017) goes on to propose the use of an SVM classifier with wavelet-based features, achieving a precision and recall of 84.0 on the task of elephant rumble detection. Leonid and Jayaparvathy (2022) applied a CNN model to public domain data, some of which we shall also use in our experimental evaluation. Achieving an average recall of 94.4, a specificity of 88.2 and a precision of 89.2.

The work described so far has made use of manual field recordings. Passive acoustic monitoring (PAM), an alternative approach to data collection, has become a popular means to record large audio datasets autonomously (Wrege et al. 2017), and has enabled the use of more sophisticated classification models. To our knowledge, Bjorck et al. (2019) were the first to apply a deep neural network to automatic elephant call detection to PAM recordings. Using MFCC features and a CNN-LSTM classifier, the authors achieve a precision of 90.8 and recall of 96.4.

Table 2. Summary of elephant voices (EV) and linguistic data consortium (LDC) elephant vocalisations datasets used for experimentation.

Dataset	EV	LDC
Authors	Poole and Granli (2021)	de Silva (2010b)
Elephant species	<i>Loxodonta africana</i>	<i>Elephas maximus</i>
Recording environment	Handheld field recordings	Handheld field recordings
Recording equipment	ARES-BB Nagra	Fostex FR-2
Microphone	Not specified	Earthworks QTC50
Number of call types	27	7
Detail of annotation	File-level	Within 100 ms
Total recording duration	1 hour	57.5 hours
Of which annotated	36 minutes ^a	5.4 hours
Number of recordings	226	1577
Average length	14.75 s	131.03 s
Min.	0.49 s	1.25 s
Max.	296.52 s	3889.38 s
Std. dev.	31.48	177.02 s
Number of vocalisations	514	4433
Sampling rate	44.1 kHz	16 kHz
Bit depth	16-bit	24-bit
Number of channels	2	1 ^b
Low frequency cut-off	Not specified	3Hz

^aThe second channel is used for field notes.^bAnnotations provided by authors, as described in Section 5.1.

5. Data

Our experiments were conducted on two audio datasets that contain elephant vocalisations. The first is a compilation of handheld field recordings of the African elephant, available in the public domain through the *Elephant Voices (EV)* conservation project (Poole and Granli 2021). The second is a collection of handheld field recordings of the Asian elephant, provided by the *Linguistic Data Consortium (LDC)* (de Silva 2010b). Both datasets are of free-roaming elephants. Table 2 provides a high-level comparison of the two datasets.

Figure 1 shows three call exemplars from the EV dataset. The spectrograms illustrate complex acoustic call structure that varies over time, that contains both harmonic and non-harmonic elements.

5.1. Elephant voices project (EV dataset)

Over the course of several years, Poole and Granli (2021) have collected a set of recordings of African elephant (*Loxodonta cyclotis*) vocalisations. The recordings were made in three locations: Amboseli and Maasai Mara in Kenya, and Gorongosa National Park in Mozambique. A subset of these recordings (approx. 230) is available in the public domain. This subset has been annotated according to an ethogram presented in Poole et al. (1988) and Poole (1994). Each recording is accompanied by a single elephant call type annotation indicating the dominant or overarching call type. However, multiple different calls may be present in a recording. This data can therefore be regarded as *weakly* labelled since there is only a single label per recording, and there is no temporal information indicating the exact time of occurrence of the call. To address this, we have added temporal labels to this dataset by manually identifying the start and end of each elephant

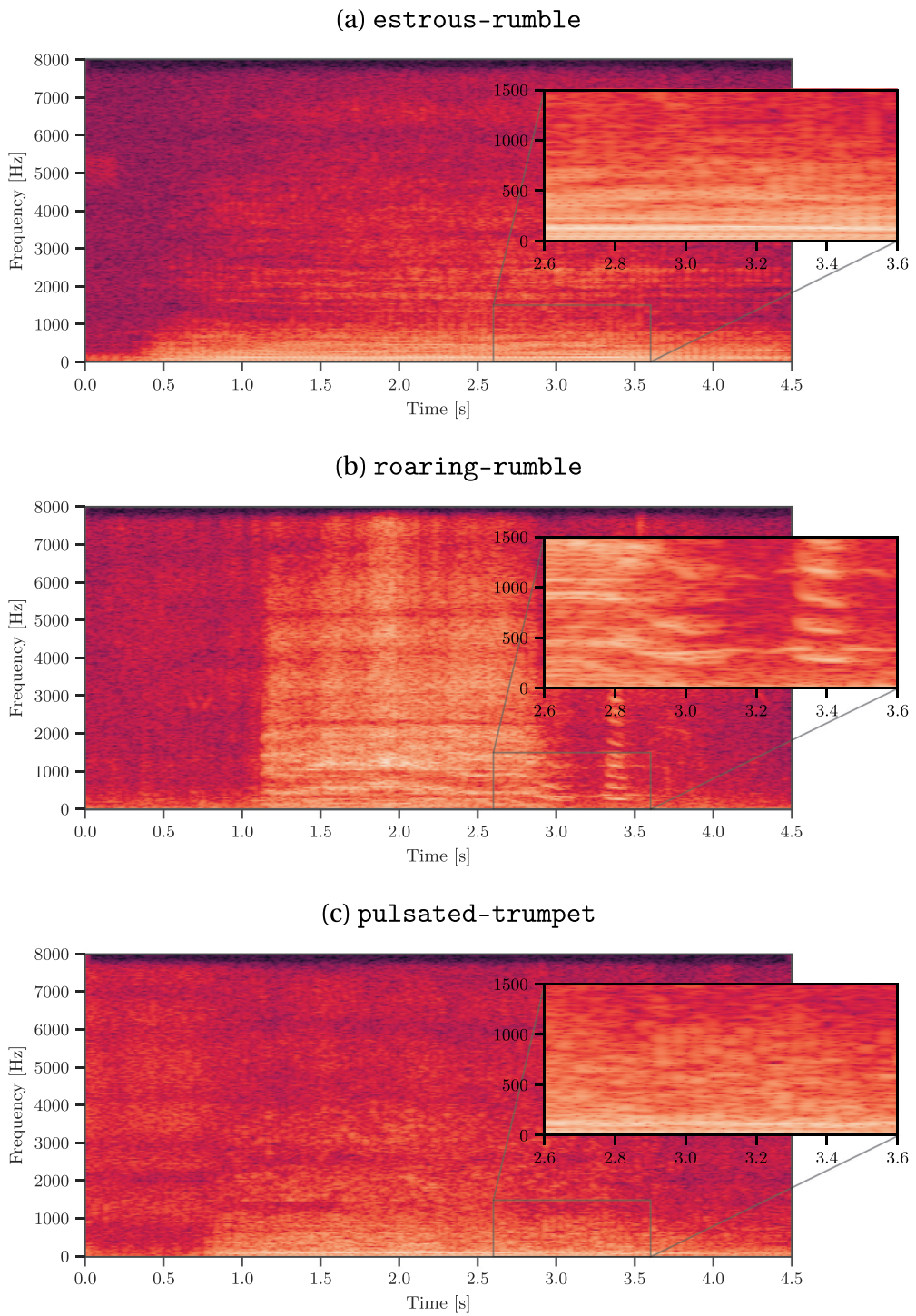


Figure 1. Log-scaled power spectral representation of three different elephant call exemplars drawn from the Elephant Voices (EV) dataset. The inset for each subfigure shows a magnified view of the low frequency spectral content.

vocalisation. The addition of these *strong* labels allows the dataset to be used not only for elephant call classification but also for call detection and endpointing. The resulting time-labelled dataset contains 922 unique vocalisations spanning 36 min of audio in 230 recordings of the total 1-h recording duration. All recordings were made using an ARES-BB Nagra recorder. However, the model of the microphone used is not specified. This dataset is similar to the data used by Leonid and Jayaparvathy (2022), thus making results the reported comparable to the findings in their study.

5.2. Asian elephant vocalisations (LDC dataset)

The Linguistic Data Consortium (LDC) hosts a dataset of Asian elephant (*Elephas maximus*) vocalisations recorded and annotated by de Silva (2010b). The dataset consists of 57.5 h of Asian elephant vocalisation recordings, made in the Uda Walawe National Park, Sri Lanka; of which 31.25 h have been annotated. The vocalisations are identified based on those that show clear fundamental frequencies (periodic), those that do not (aperiodic) and those that show both periodic and aperiodic regions. In this way, seven unique call types can be identified (McKay 1973). The dataset is strongly labelled and includes the diarisation for up to four overlapping elephant vocalisations. To allow accurate diarisation, the dataset is annotated in a *multi-label* fashion, which allows more than one call type to occur at the same time. Field recordings were made using a Fostex FR-2 field recorder and an Earthworks QTC50 microphone (capable of recording infrasonic signals down to 3 Hz).

5.3. Call annotations

The EV and LDC datasets have been annotated using different labelling schemes (McKay 1973; Poole et al. 1988). These differences not only relate to the terminology used by the authors but also to intrinsic differences in the ecological behaviour and acoustic properties of the vocalisation of the two different elephant species. Langbauer (2000) presents a unified annotation scheme for elephant vocalisation and compares the differences between annotation styles used in the two datasets we consider. Table 3 summarises the different call types in each of the two datasets according to this unified annotation scheme, and specifies total call duration and number of occurrences, for each dataset, respectively.

Each dataset does not contain the same range of call types. However, both corpora contain rumble, trumpet and roar, allowing for comparison of classifier performance across datasets for these call types. While both corpora contain croak and squelch, there are insufficient samples of these call types to reliably evaluate classifier performance. Furthermore, the snort, nasal-trumpet, husky-cry and truck-like call types have been omitted, due to an insufficient number of occurrences to allow cross-validation.

Figure 2 shows the prevalence for each respective call type in each corpus. From this figure, it is clear that both datasets are severely unbalanced, with the rumble call type dominant in both, followed by roar and squelch in the LDC dataset and trumpet and roar in the EV dataset. The rumble and roar call type occurs with approximately the same frequency in both datasets. To address this class imbalance, performance metrics will be

Table 3. Call types present in the elephant voices (EV) and the linguistic data consortium (LDC) corpora, according to the unified annotations convention proposed by Langbauer (2000). The number of segments for each call type, the total duration of each call type, and the associated target label are listed. The number of cross-validation folds is denoted by K and is smaller for EV due to its smaller size.

Langbauer (2000) call type	EV ($K = 5$)		LDC ($K = 10$)	
	# Segments	Total duration [sec]	# Segments	Total duration [sec]
Advertisement of hormonal state				
Oestrous rumble	40	169.7	—	—
Musth rumble	6 ^(b)	34.2	9 ^(a)	11.0
Advertisement of emotional state				
Female chorus	23	198.8	—	—
Long roar	—	—	150	413.3
Roar	19	42.1	47	76.0
Mating pandemonium	18 ^(b)	177.8	—	—
Play trumpet	9 ^(b)	7.5	—	—
Social trumpet	4 ^(a)	6.4	—	—
Trumpet blast	27	49.8	—	—
Snort	6 ^(b)	7.6	—	—
Trumpet	28	54.4	144	195.9
Rumble	162	695.6	183	1234.2
Chirp-rumble	—	—	23	77.9
Croak-rumble	—	—	11	19.8
Group cohesion and coordination				
Growl ^c	—	—	2972	15476.7
Let's go rumble	6 ^(b)	35.5	—	—
Contact rumble	40	138.8	—	—
Greeting rumble	40 ^(b)	178.7	—	—
Affiliative				
Cry	9	13.17	—	—
Misc				
Squelch	6	11.5	43	53.4
Squeak	—	—	423	250.5
Croaking	5 ^(b)	29.4	—	—
Woosh	4(a)	6.6	—	—
Bark	—	—	35	35.4
Trunk call	8	23.8	—	—
Ceremony	18	227.4	—	—
Composite				
Long roar-rumble	—	—	254	1074.4
Roar-rumble	5	15.4	75	255.2
Bark-rumble	—	—	142	450.3
Rumble-cry	6	10.0	—	—
Total	489	2134.2	4511	19624.1

^aOmitted due to fewer than one segment per cross-validation fold.

^bOmitted due to fewer than one recording per cross-validation fold.

^cBased on the findings in de Silva (2010a), the Asian elephant's *growl* is acoustically similar to the African elephant *rumble* identified by Poole et al. (1988).

computed for each class independently and the average over all classes taken. There is also a considerable difference in the number of segments, the LDC dataset containing at least an order of magnitude more segments for each call type.

5.4. Composite calls and subcalls

Elephants also produce a series of amalgamated calls, referred to as *composite* calls (Poole 2011). These calls consist of a combination of two or more fundamental call types (e.g.

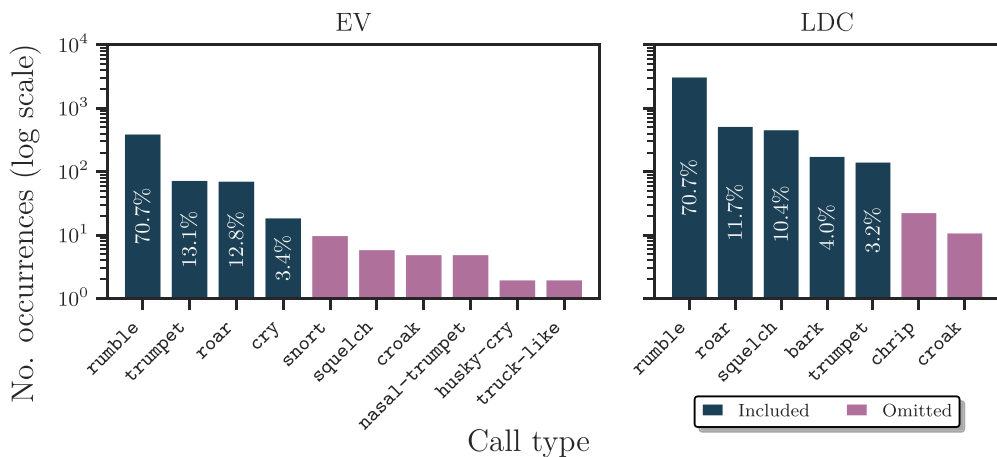


Figure 2. Number of occurrences of each call type in the Elephant Voices (EV) and the Linguistic Data Consortium (LDC) corpora. Some call types are omitted from experiments due to an insufficient number of occurrences to allow cross-validation. Each included call type has been annotated with the class prevalence of each call type, within their respective dataset.

rumble and roar) produced as a single vocalisation (e.g. roar-rumble). Composite calls have received limited attention in literature, but have been observed to indicate excitement or disturbance (Poole 2011). For the purpose of our experiments, these calls have been annotated based on their fundamental call types, for example, the start of a roar-rumble would be annotated as roar, while the end would be annotated as rumble without any separating silence. As the vocalisation transition is seamless between the different call types, the exact transition boundary may be difficult to annotate exactly.

Elephants do not only exhibit vocalisations that form part of their major call types, such as rumble and roar, but also have more nuanced calls referred to as *subcalls* (Poole 2011). These calls accompany particular elephant behaviour during, for example, parent-offspring interactions, hormonal and emotional state and social interaction. As this behaviour cannot be determined based only on the major call types, automated subcall classification is the first step towards automated non-invasive elephant behavioural classification. We perform subcall classification experiments only for the EV dataset, as the LDC data do not contain subcall annotations. Figure 3 shows the spread of the subcalls types used for experimentation. Some subcalls are too infrequent to perform cross-validation and have thus been omitted.

5.5. Cross-validation

When developing machine learning-based models, separate training, testing and development datasets are required. Since our datasets are small for most vocalisation classes, we have partitioned our data into training, testing, and development datasets with a view to cross-validation (Mosteller and Tukey 1968; Stone 1974).

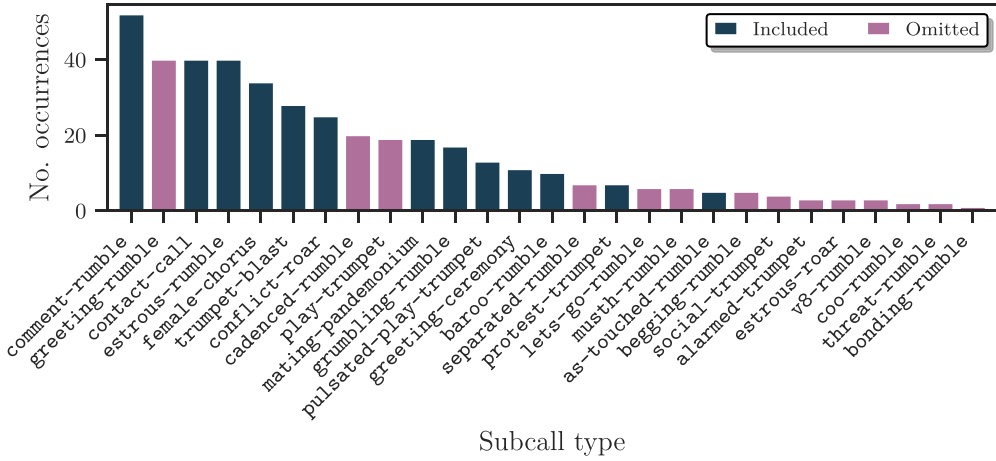


Figure 3. Number of occurrences of each subcall type present in the Elephant Voices (EV) corpus. Some subcall types are classes omitted from experiments due to an insufficient number of occurrences to allow cross-validation.

K -fold cross-validation partitions the data into K disjoint subsets referred to as *folds*. In a sequence of K *turns*, each of the folds is held out as a test set, while the remaining $K - 1$ folds are used for model training.

Nested cross-validation extends this process by introducing an *inner* and *outer* turn. For each outer turn, a test fold is kept aside for testing. For each inner turn, from the remaining $K - 1$ training folds a development fold is held out for parameter selection and model performance validation while training the model on the remaining $K - 2$ folds. The best performing model, based on the $K - 1$ inner fold development sets, is chosen and retrained on the complete inner fold while still excluding the outer (test) fold. Hence, for each inner turn, a different fold is used as a development set. The optimal hyperparameters for each outer turn are selected based on the average inner turn development loss.

When dividing a dataset into folds (subsets), it is important to ensure that the class distribution remains even. Stratification is the process of ensuring the target label distribution for a given partition of the data is representative of the overall dataset target label distribution. The dataset is partitioned such that each fold is stratified with respect to the number of elephant call types, thereby preserving the overall call occurrence distribution within each fold. Additionally, no single recording occurs in more than one fold.

Cross-validation is especially suitable for small datasets because here the risk of choosing a single static test set that is biased due to sampling variation is higher than it is for large datasets. Both datasets, in Table 2, are small in comparison to those typically used in audio classification (Gemmeke et al. 2017) and sound event detection (Serizel et al. 2020) tasks. Therefore, nested K -fold cross-validation with stratified class sampling has been used throughout. We employ 5-fold and 10-fold cross validation for the EV dataset and LDC datasets, respectively.

5.6. Data pre-processing

The following section describes how the raw audio was pre-processed before it was used for model training and evaluation.

5.6.1. Resampling and input normalisation

Since some recordings span several minutes, each recording was first divided into shorter intervals while ensuring no call was split. Furthermore, all audio recordings are resampled to a common 16 sampling rate. The *LDC* dataset has two audio channels, the first containing the animal sound recording and the second containing voice memorandums with field notes made during the recording. Only the first channel was used in our experiments. The *EV* dataset also has two audio channels, in this case a stereo recording. These channels were averaged to produce a single channel recording. Finally, all recordings were normalised to have zero mean and a peak amplitude of -1 dB.

5.6.2. Feature extraction

Spectral-temporal feature representations, in the form of mel-frequency spectra and mel-frequency cepstra were computed from the resampled audio signal, using a 25 ms frame length (S_l), a stride (S_s) of 10 ms and a Hamming window function. While these values are common in speech processing tasks, their efficacy has yet to be shown for elephant vocalisations. We experimentally evaluate different frame lengths S_l and inter-frame stride S_s , for a set of shallow classification models. However, some of the pre-trained deep-neural models use a standardised frame length and inter-frame stride. In such cases, this configuration was maintained across models, to allow the direct comparison between model architectures and to allow for the application of transfer learning.

For mel spectrograms, a frame-wise spectral representation is obtained by computing a 1024-point dft from a zero-padded and windowed frame. The magnitude of the complex-valued discrete spectrum is squared to obtain the real-valued power spectrum. The power spectrum is mapped to the mel-frequency scale using a bank of N_m log-spaced triangular filters. The resulting mel-scale spectrogram can be further transformed to a mel-frequency cepstrum representation by taking the logarithm of the mel-scale power spectrum and applying the discrete cosine transform. Of the resulting N_m MFCCs, the first N_c coefficients are retained. MFCCs features were only evaluated for shallow classification models. The value of N_m and N_c , is both considered to be experimental hyperparameters.

6. Call detection, endpointing and classification

Call activity detection is the determination of whether an elephant vocalisation, of any call type, occurs in a certain audio signal. We will refer to the task of *call activity detection* simply as *call detection* throughout. These audio signals can be highly *polyphonic*, in other words, contain multiple other audio sources besides elephant vocalisations. These additional sources may be biological (e.g. birds) or mechanical (e.g. vehicles). *Call endpointing* is the task of isolating the beginning and end of a call, within an audio

signal. When call detection is performed at discrete time intervals for an audio signal, endpointing is implicitly also performed.

We first define a *frame* as a short, regularly spaced, interval of an audio signal, from which a *feature vector* is obtained. The length of the frame is denoted by S_f and the stride between successive frames by S_s (Section 5.6.2). Furthermore, we define a *context window* as a collection of consecutive feature vectors in time. The length of the context window is denoted by W_l . The context window serves as the input to a classification model, resulting in a *single* classification output probability. By shifting the context window in time using a fixed stride W_s , a series of classification probabilities is obtained. Therefore, the context window stride W_s determines the temporal resolution at which the call detection and endpointing is performed.

When the feature vector frame length (S_f) is equal to the length of the context window ($S_f = W_l$), it means that only a single feature vector is presented to the model as input. When the context window length is longer than the feature vector frame length ($W_l > S_f$), it means that a series of feature vectors is presented to the model.

We further define a *segment* to be an interval of audio that has been endpointed to contain a particular call. Figure 4 shows an audio signal containing two segments along with the process of extracting frames, computing feature vectors and composing context windows.

Call classification is the identification of the type of vocalisation (rumble, trumpet, etc.) associated with a call isolated during endpointing (segment). We will regard call classification as a sequence classification task, where a single classification label is associated with the entire segment. This is in general a multi-class multi-label classification problem because there are more than two call types (multi-class) and more than one call may occur simultaneously (multi-label).

In the above description, call detection is described as a binary classification problem (call/no-call). It is, however, also sometimes considered as a multi-class multi-label

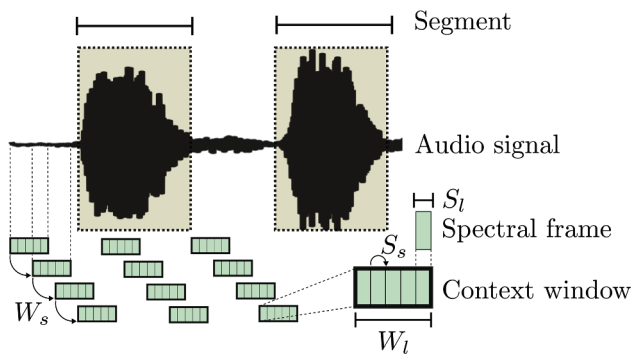


Figure 4. Illustration (not to scale) of the process of constructing a context window consisting of a collection of feature vectors from an audio signal. Shown is the context window, for which the length in the number of audio samples is denoted by W_l and the stride by W_s . For each context window, a series of features vectors is obtained using a frame length of S_f and a stride of S_s . This sequence of feature vectors (context window), serves as the model input. Note that, while the frames from which the feature vectors are obtained, are shown here to be non-overlapping, in practice these frames may overlap ($S_f > S_s$).

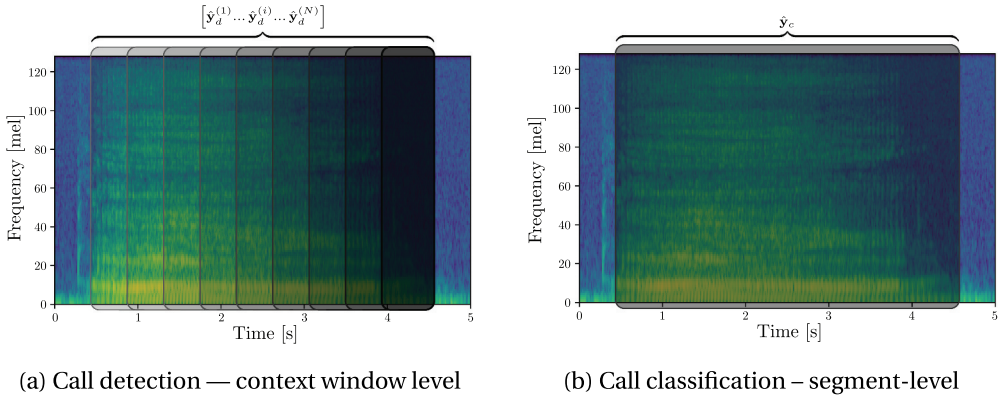


Figure 5. Mel-spectrogram feature representation of an elephant call used to illustrate **(a)** call detection and **(b)** call classification. On the left, $\hat{y}_d^{(i)}$ denotes the detection output for the i -th time instant, extracted from one recording. The shaded area indicates the context window for which a positive detection decision was made. On the right, \hat{y}_c denotes the multi-label classification output for a single elephant call, already endpointed. Here, the shaded area denotes the endpointed segment.

classification problem, where not only the presence and location in time of the call is determined but also its identity (rumble, trumpet, etc.). From here on we shall refer to multi-class multi-label call detection simply as multi-label call detection.

Figure 5 illustrates the difference between call classification and call detection.

7. Experimental setup

In this section, we describe the procedure used to train and evaluate the classifiers described in Section 3.

For binary call detection, we follow a similar approach to Zeppelzauer et al. (2013). Unless otherwise stated, we apply each model to a context window (consisting of one or more feature vectors) and compare a single multi-label output. Subsequently, the context window is shifted in time, and the next model output is computed.

For call classification, an oracle model is assumed to have segmented the call, meaning that the endpoints are obtained from the ground-truth annotations. The classifier is then applied to this segment of the audio signal. This results in several multi-label classifications probabilities for each time instant in a single call segment (Section 6). We compute the average² of these classifications probabilities over time to determine the final classification result for the segment. For model architectures that support multi-label output, such as random forest and MLP with a sigmoid output activation, no further modifications to the models are required. In the case of model architectures that do not, such as logistic regression, we train C binary classifiers, one for each of the C classes. In this work, we regard the multi-label call classification to be a down stream task of call detection and endpointing. Thus, results presented for call classification are assumed to be down stream from an oracle call detection system. We evaluate call detection using both a binary (e.g. call/no-call) and

multi-label (any of rumble, trumpet, etc.) schemes, while we always regard the task of call classification in a multi-label scenario.

7.1. Call detection

We perform call detection using two different strategies: (1) applying a classification model to a context window repeatedly over time and (2) using a one-to-one sequence classification model, both illustrated in Figure 6.

The first strategy takes as input a context window $\mathbf{X}^{(i)}$ composed of one or more sequential features vectors $X^{(i-\frac{W}{2})} \dots X^{(i+\frac{W}{2})}$. The feature vectors themselves are obtained through the spectral feature extraction methods described in Section 5.6.2. The classifier produces a single classification output $\hat{y}_d^{(i)}$ associated in time with the i -th (centremost) classification interval of the input context window. Experimentally, we evaluate different input context window lengths W_l , while keeping the classification resolution fixed ($W_s = C_s = 100\text{ms}$), allowing for comparable results (Section 7.1.1).

The second strategy takes as input a longer context window $\mathbf{X}^{(i,k)}$. We refer to this longer³ context window as an *extended context window* in the text for clarity. Instead of producing a single classification output, the sequence model produces a series of classification outputs $[\hat{y}_d^{(i)} \dots \hat{y}_d^{(k)}]$ associated with classification time instants i to k .

7.1.1. Target classification labels

In order to obtain target labels for the call detection task, the annotation intervals (start and end times) have to be discretised to match the classification resolution, as described in Section 6. Since both EV and LDC datasets contain overlapping annotations (Section 5.3), due to multiple elephant vocalisations occurring simultaneously, the target labels will be encoded in a multi-label fashion.

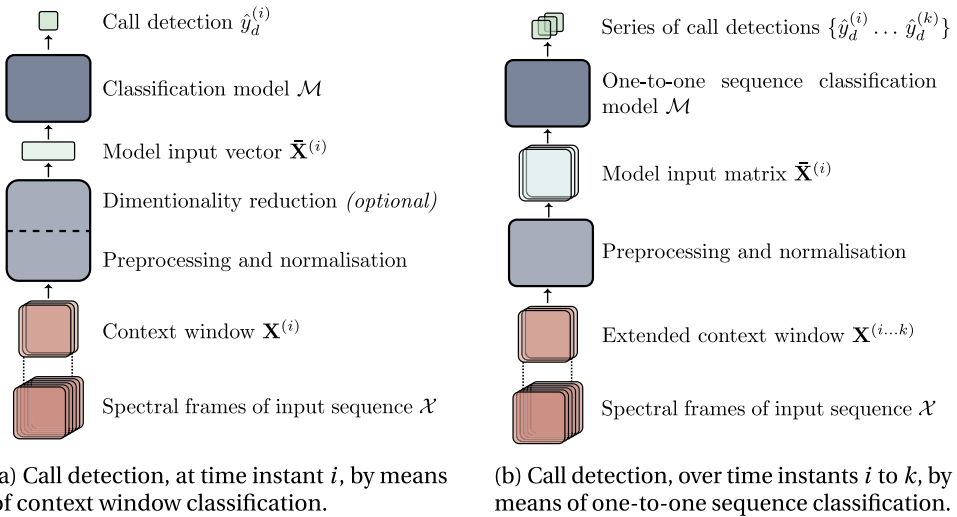


Figure 6. Model overview of classification models used for call detection. Only a single input sequence is shown, and batch processing is omitted from the illustration.

Each classification target consists of a vector with a dimensionality equal to the number of call types present in the task, with an additional label reserved to indicate that no call is present (not-call). Each dimension in the target vector corresponds to a binary value indicating the presence or absence of that particular call type. This process is illustrated in Figure 7.

The annotation intervals for each recording are discretised as follows: First, a series of regularly spaced classification vectors are constructed for each recording, each associated with the output of the classifier at a different instant. The spacing between these vectors are denoted by C_s . Because C_s determines the target classification resolution, it should be equal to the context window stride ($C_s = W_s$). For each classification vector, a short interval (C_l) of the annotation is considered. An annotation label must occupy at least 20 of the classification interval (C_l) to be considered a valid label for that classification instant. When no other label is associated with such an instant, the no-call label is assigned. Figure 7 illustrates the process employed to generate the discrete classification targets over time.

The target classification resolution has been fixed to 100 ms for all experiments, ensuring that all results are directly comparable. This is achieved by setting $W_s = C_s = 100ms$. We also fix the length of the classification interval to $C_l = 100ms$.

This classification scheme allows the model input parameters to be changed without affecting the classification interval. In other words, the context window length W_l and feature vector extraction frame length S_l and stride S_s can be chosen independently of the classification interval and allows for these parameters to become experimental hyperparameters.

This configuration also allows for context windows to extend beyond the classification interval ($W_l > C_l$). In such cases, the classification target is associated with the centre of the context window.

Finally, when scoring, the start and end of recordings are truncated according to the model with the overall longest context length. This allows for the direct comparison of results between models.

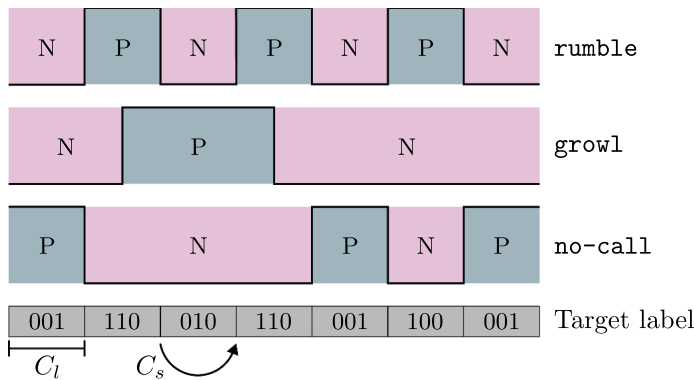


Figure 7. Illustration (not to scale) of the target label generation process, for two call types and the no-call class. The discrete classification interval length is denoted by C_l and stride by C_s .

7.1.2. Shallow classification models

As shown in Figure 8a, shallow classification models produce a single output estimate per input context window. The context window is shifted in time according to the classification stride (100 ms), after which the next classification is made. This results in a series of classification results for a given audio signal.

We evaluate the shallow models both when using a single spectral frame as input and also when providing several concatenated spectral frames (context window). The length of the context window (number of concatenated frames) is treated as an experimental hyperparameter. Such concatenation can lead to high-dimensional input vectors, which may in turn lead to overfitting or poor convergence during training. We therefore additionally evaluate principal component analysis (PCA) as a dimensionality reduction step. This dimensionality reduction as well as cepstral mean and variance normalisation are also considered as experimental hyperparameters.

7.1.3. SVM probability calibration

Certain models, such as the SVM, do not produce a class membership probability as output. Instead, these models provide a classification score. Probability calibration, in the context of classification models, is the process of using an additional classifier or regression model to map these classification scores to a class probability. We considered two methods of achieving this. The first method is Platt scaling (Platt 1999), which adds a logistic regression classifier to the output of the SVM to produce the desired target label probability. The second method is isotonic regression (Zadrozny and Elkan 2001, 2002), which fits a piecewise-constant non-decreasing function, to the output of the decision function. Isotonic regression was chosen as it has shown to offer better performance than Platt scaling, especially when sufficient training data are available (Niculescu-Mizil and Caruana 2005).

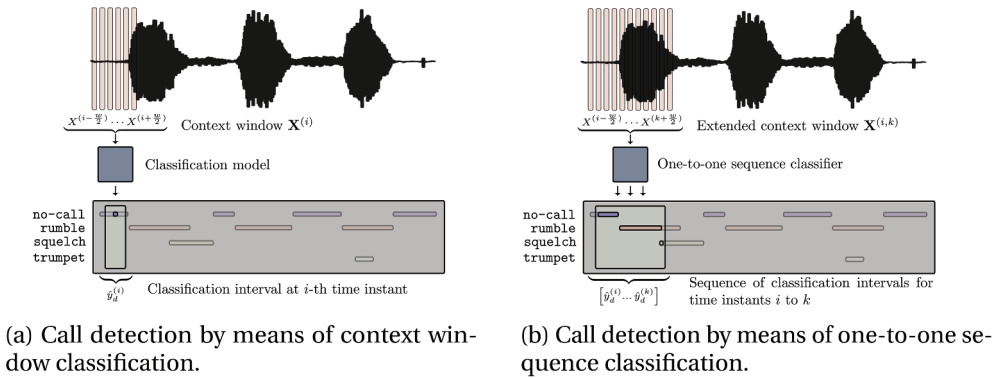


Figure 8. Illustration of the two strategies followed for call detection, described in Section 7.1. On the left, $\hat{y}_d^{(i)}$ denotes the classifier output for the i -th (centre) input frame, given the input context window $X^{(i)}$ consisting of w consecutive spectral features $X^{(i-\frac{w}{2})} \dots X^{(i+\frac{w}{2})}$. On the right, $[\hat{y}_d^{(i)} \dots \hat{y}_d^{(k)}]$ denotes the classification sequence produced by the classifier for frames i to k , given an extended context window $X^{(i,k)}$.

7.1.4. Kernel approximation

In the case of SVM classifiers, we experimentally evaluate the use of different non-linear kernel functions. In order to compute the exact kernel function requires on the order of $\mathcal{O}(N_s^3)$ computations. This cubic scaling in the number of input samples (N_s frames) is computationally intractable for the datasets used in this work. Thus, we consider two methods for approximating the kernel function, the first is using *Nyström* method (Williams and Seeger 2000) and the second is *Random Kitchen Sinks* method (Rahimi and Recht 2008). The particular kernel function and kernel approximation methods used are left as an experimental hyperparameter. However, we evaluate both the radial basis function and a third degree polynomial as a basis for the kernel function.

7.1.5. Convolutional neural network

Beyond shallow classification models, we also evaluate a series of deep convolutional classifier architectures, including AlexNet (Krizhevsky et al. 2012), ResNet (He et al. 2016), and VGGNet (Liu and Deng 2015). Unless otherwise specified, all deep architectures were trained using the *Adam* (Kingma and Ba 2014) optimiser with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ with binary cross entropy as the loss function. The initial learning rate of the optimiser is considered a hyperparameter.

To ensure that the model is adequately trained, we monitor both the training and the development loss for convergence, or divergence. If either loss diverges (increases) for more than three epochs or converges (remains unchanged within a threshold), the training process is terminated. In the case of divergence, we rewind the model parameters to the previous state with the lowest developmental loss. For AlexNet and ResNet architectures, we also test whether transfer learning on an out-of-domain image dataset (e.g. ImageNet) improves performance compared to random initialisation of the model (Deng et al. 2009).

Like the shallow classifiers, the convolutional models take a temporal-spectral feature representation (mel-spectrogram) as input. The exact configuration of this mel-spectrogram is discussed in Section 5.6.2. Context window lengths in the range 0.5 s to 3 s are considered experimentally, but in all cases, a single classification output associated with the centre 100 ms interval of the context window is produced. Hence, like the shallow architecture, we use CNNs to achieve call detection by means of context window classification, as illustrated in Figure 6a.

We evaluate the standard model configuration for AlexNet (61.1 params.), while for VGGNet we evaluate four variations of different size, the largest of which contains 143.6 parameters. While VGGNet was considered, it lead to no noticeable performance increase over AlexNet, and is thus omitted from the results. We consider the five standard resnet configuration with the smallest, *resnet-18*, containing 11.7 M parameters and the largest, *ResNet-152*, containing 60.2 M parameters.

7.1.6. Transformer-encoder

Finally, we evaluate the transformer-encoder as a one-to-one sequence classification model for call detection, as shown in Figure 6b. As described earlier, unlike the context window classification models, which produce a single classification probability per input context window, these sequence model architectures produce

as many classification outputs as there are input sequence tokens (patches). As the classification resolution of the model is determined by the model configuration, the extended context window stride does not have to be fixed to $C_s = 100\text{ms}$. As such there can be no to minimal overlap between extended context windows, leading to greater efficiency. Such models also have the ability to model temporal dependency and make estimations based on previous, and in some cases future, observations, by means of the self-attention mechanism. [Figure 6](#) provides an illustration of such sequence classification models, compared to context window classification models.

We choose the AST one-to-one sequence classification model for experimentation, described in [Section 3.2.6](#). We evaluate two variants of the AST model: AST-seq and AST-cls.

In the first variant, AST-seq, we omit the learned classification token [CLS] that is usually prepended to the input sequence, and instead we train an outer MLP classification layer directly on the output representation of the transformer model ([Chen et al. 2022](#)). As the model produces 2D spectral-temporal representation as output, we average the spectral features to obtain a single temporal representation per time step.

For encoder-only transformer models, there exists a one-to-one correspondence between the number of input and output tokens (patches). Specifically, for an input sequence consisting of N tokens, the model will produce N outputs ([Section 3.2.6](#)). In the case of the AST architecture, each input sequence token consist of 16 mel frequency bins drawn from 16 consecutive spectral frames (16×16 patches). Thus, a classification estimate is produced approximately once every 160 ms (for a frame stride of 10 ms). This model classification rate does not exactly match the classification rate of one classification per 100 ms used in our other experiments. To address this discrepancy, the classification output is resampled using linear interpolation. However, this resampling process is not included during model training, only during scoring.

In addition to the AST-seq variant, we also evaluate the standard AST model architecture as proposed by the [Gong et al. \(2021\)](#), which uses the learned classification token [CLS] from which the single call classification probability is produced. This variant of the model is referred to as AST-cls. Note that while AST-seq is a sequence classification model producing a series of classification probabilities for a given extended context window, AST-cls is a traditional classification model that produces on a *single* output per context window.

Furthermore, we experimentally evaluate the effectiveness of training the AST model from a random initialisation, as well as pre-training the model using the self-distillation scheme presented by [Chen et al. \(2022\)](#) on AudioSet ([Gemmeke et al. 2017](#)). During fine-tuning, we employ a training regime, similar to that proposed by [Vasconcelos et al. \(2022\)](#), in order to preserve semantic information obtained during pre-training. That is, for the first 10 epochs of training, the backbone feature extraction neural network is fixed (no gradient update is performed). After this, the backbone network is updated with a learning rate that is 1 of the outer MLP classification layer's learning rate. The backbone learning rate is then increased over the next two epochs to match the overall learning rate. This technique is common practice in fine-tuning models as it reduces the chance of losing transferable semantic information learned during pre-training and has been showed to improve performance on classification tasks ([Vasconcelos et al. 2022](#)). The

model's parameters are optimised using the *AdamW* (McInnes et al. 2018) optimiser with $\beta_1 = 0.9$ and $\beta_2 = 0.98$, and a weight decay factor of 0.01.

7.2. Call endpointing

Call endpointing refers to the specific task of locating the precise beginning and end of an elephant vocalisation within a continuous audio recording. The resolution of the call endpointing is determined by the classification resolution of the model ($msC_s = 100ms$). By accurately identifying the start and end times of elephant calls, the efficiency and accuracy of subsequent data processing tasks is enhanced.

Endpointing is performed implicitly in this work, through binary call activity detection as described above. In order to achieve endpointing, a threshold (e.g. $\theta = 0.5$) is applied to each of the call detection probabilities over time to classify each time instant (context window) as a positive (call, presence) or negative (not-call, absence). The start of a segment (new call) is identified by a negative call detection followed by a positive (rising edge), the end of segment (call) is identified by a positive call detection followed by a negative (falling edge).

7.3. Call classification

Call classification aims to determine which calls are present in a given audio segment. The task of call classification seeks to determine which of the major or subcall types a particular audio sequence belongs to. In sound classification tasks, it is typically assumed that there is a single dominant acoustic source present and that the audio is already endpointed – i.e. the start and the end of the call are also the start and end of the presented audio segment. This implies that there is no temporal context indicating for where in the overall recording the call occurs.

In order to perform call classification, we employed the same models used for call detection and therefore also the same selection of hyperparameters. Each segment is divided into several context windows, for which the classification model produces a single classification per context window. This results in a series of call detection probabilities over time for each call segment. Call classification is achieved by averaging the call detection probabilities over the endpointed segment. This average is considered the output of the call classifier models.

As before, the target label for each segment is obtained from the annotations. However, a single multi-label target output is assigned to each call segment to be classified. In cases where neighbouring calls overlap with a given segment, the neighbouring call must occupy at least 50 of the call segment to be considered present. This does, however, mean that the interval between overlapping calls are evaluated twice, once for the first segment and again for the succeeding segment.

7.4. Performance evaluation metrics

Several performance metrics will be used to assess the ability of the models to perform call detection, endpointing and classification, as described in the previous sections.

7.4.1. Dealing with class imbalance

The datasets utilised in this study exhibit considerable class imbalance, an issue that is prevalent in many animal vocalisation studies (Section 5.3). This disparity may distort performance assessments based on certain metrics. To mitigate the influence of class imbalance on multi-label classification results, we implement two **scoring** strategies: one-vs-one scoring and macro-averaging across all classes.

7.4.1.1. One-vs-rest classification. In multiclass scenarios, binary classification metrics can be applied by considering comparisons between a target class (positive) and all the other classes (negative). This approach produces a set of scores for each classification category and is referred to as one-vs-rest classification. However, one-vs-rest metrics remain vulnerable to class imbalance (Bishop and Nasrabadi 2006; Lorena et al. 2008).

7.4.1.2. One-vs-one classification. A variation of one-vs-rest classifiers is the one-vs-one approach, in which a single target class is contrasted with a single different class, for instance, *rumble-vs-trumpet*. One-vs-one scores demonstrate resilience to class imbalance when employing robust metrics. Since all classifiers presented in this paper include an explicit no-call class, we evaluate each class as a binary classifier versus the no-call class (e.g. *rumble-vs-no-call*).

7.4.1.3. Multi-class macro-averaging. Certain classification metrics, such as classification accuracy, may suffer from misleading results if the dataset is unbalanced. One way of overcoming this is to compute the metric for each respective class (*one-vs-rest* or *one-vs-one*) and then compute the average. Since each individual metric is normalised within its class, this avoids skew. Using one-vs-one, as individual metrics has been shown to be more robust to class imbalance (Bishop and Nasrabadi 2006; Lorena et al. 2008).

7.4.2. Classification metrics

Call classification performance was assessed using the binary metrics: sensitivity, specificity, precision and recall, and derived quantities. All performance metrics reported are calculated separately for each outer cross-validation fold and then averaged. We consider if a particular call type is present as a positive class and vice versa. We treat each separate class (call type) as its own binary classification problem (multi-label).

7.4.2.1. Specificity and sensitivity. Specificity measures the true negative rate, i.e. the proportion of true negatives among all actual negatives, given by:

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{TN}{N} = 1 - FPR \quad (1)$$

Sensitivity, also known as recall, measures the proportion of true positives among all actual positives, given by:

$$\text{Sensitivity} = \text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{P} = TPR \quad (2)$$

A high specificity implies a low false positive rate (*FPR*), while a high sensitivity implies a high true positive rate (*TPR*).

7.4.2.2. Receiver operating characteristic. Specificity and sensitivity reflect classification performance at a single operating point (i.e. decision threshold). Increasing the decision threshold will result in an increase in the system's sensitivity, at the cost of a decrease in its specificity, and vice versa. The receiver operating characteristic (ROC) is to visualise the trade-off between system specificity and sensitivity. An ROC curve is drawn by plotting a locus of *TPR* against *FPR* for a series of decision thresholds.

7.4.2.3. AUC ROC. The area the ROC curve (AUC ROC) provides a single-figure indication of classifier performance across all decision thresholds. An AUC ROC score of 1.0 indicates a perfect classifier, whereas a score of 0.5 indicates the classification performance achieved by guessing the prior probability (i.e. a random classifier).

7.4.2.4. Precision and recall. As alternatives to specificity and sensitivity, two other commonly used metrics are precision and recall. Recall is equivalent to sensitivity. Precision is the ratio of true positive (*TP*) classifications to the total number of positive classifications (*PP*) made by the model. It measures the proportion of correctly identified positive instances (true positives) out of all the instances labelled as positive (both true and false positives). For a binary classifier, precision is defined as:

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{TP}{PP} \quad (3)$$

7.4.2.5. Precision-recall curve. The *precision-recall curve* is the analogue of the ROC curve. In this case, we vary the decision threshold and obtain a locus of precision and recall values. Unlike the ROC curve, the precision-recall curve is not monotonically increasing.

7.4.2.6. Average precision. For a binary classifier, the area underneath the precision-recall curve is defined as the average precision (AP). If the score is evaluated in a multi-class scenario, the average over all classes is computed and is referred to as the map.

7.4.3. Detection and endpointing metrics

For the task of call detection, models were evaluated based on coverage and purity as well as the Jaccard index. These metrics are accepted practice in the field of acoustic segmentation (Kemp et al. 2000).

7.4.3.1. Boundary precision and recall. When applied to endpointing, the metrics precision and recall can be computed based on the number of correctly detected boundaries. In this case, recall represents the number of boundaries that are correctly detected as a fraction of all postulated boundaries, while precision reflects the fraction of postulated boundaries that are indeed correct. Using precision and recall in this manner requires the introduction of a boundary tolerance parameter. A postulated boundary that falls within this margin of a true boundary will be regarded as a *TP*. This parameter must be kept constant between experiments to ensure that the associated metrics are directly comparable.

7.4.3.2. Coverage and purity. Coverage and purity are popular metrics in the speaker diarisation field. The metric is used to evaluate how well a postulated segment is representative of the ground truth segment. These metrics pose as an alternative to boundary precision and recall, and benefit from not requiring an explicit boundary tolerance. However, due to the discrete evaluation of the metric, the interval between successive classification time instants (C_s), imposes an intrinsic scoring resolution.

For a binary classifier applied to a portion of the signal, coverage is defined as the ratio between the correct positive classifications at each time instant and the number of positive samples. Therefore, coverage is merely recall computed over an interval, and is indicative of how well the classifier is able to detect calls. The associated precision metric is known as purity, and reflects the fraction of positive classifications that are indeed correct, calculated over an interval. Purity provides insight into how precise the positive classifications are. The combination of these metrics aims to measure a models' tendency to over- or under-segment a sequence.

Over-segmentation is the result of a model that over-eagerly identifies boundaries in a sequence, thus leading to excessive fragmentation. Such models generally exhibit an increased number of false-negative classification and thus lead to a decrease in coverage, but also tend to exhibit high purity. On the other hand, under-segmentation occurs when decisions are overly conservative, leading to an increase in false-positive classifications, and thus a reduction in purity while maintaining a high degree of coverage. The ideal segmentation model would have both high coverage while maintaining a high purity score. [Figure 9](#) illustrates both over- and under-segmentation.

7.4.3.3. Jaccard index. The final metric of we describe is the Jaccard index. While predominantly used in image segmentation tasks, where it is known as IoU, or numerical set similarity, it has seen application in one dimensional segmentation tasks. The Jaccard index of two sets is defined as follows:

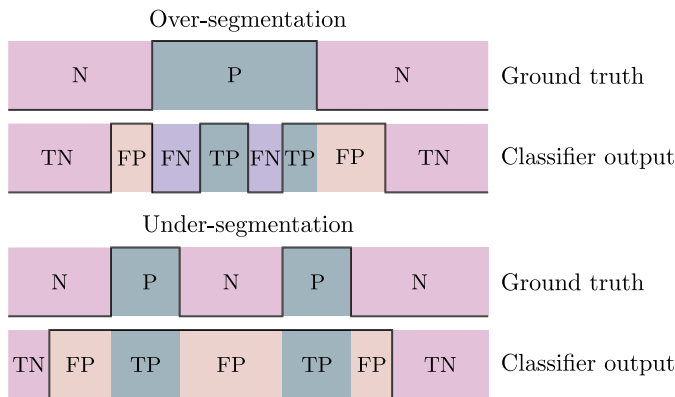


Figure 9. Illustration of over- and under-segmentation for a binary classifier. While the illustration depicts a continuous segmentation process, in this work we perform segmentation using classification labels that are discrete in time. The ground truth signal indicates call *presence* (P, positive) and call *absence* (N, negative). Furthermore, the classifier output has been annotated as *true positive* (TP), *false positive* (FP), *true negative* (TN), and *false negative* (FN).

$$J(y, \hat{y}|y_c = \hat{y}_c) = \frac{|y \cap \hat{y}|}{|y \cup \hat{y}|} \quad (4)$$

where y and \hat{y} are the ground truth and the classifier output, given that the class is the same ($y_c = \hat{y}_c$), while \cap and \cup are the intersection and union between two sets, respectively, and $|\cdot|$ is the cardinality of the set (in our particular case the duration of the classification intervals). For a binary classifier, the Jaccard index is given by:

$$J = \frac{TP}{TP + FN + FP} \quad (5)$$

The Jaccard index is a measure of the system's segmentation localisation ability. The Jaccard index is bound between 0.0 and 1.0, where 1.0 indicates a perfect segmentation model.

The Jaccard index accounts for both the coverage and purity of a segmentation model. One can thus not distinguish purely based on the Jaccard index whether a model is suffering from over- or under-segmentation. However, the Jaccard index encapsulates the overall segmentation performance as a single number.

7.5. Hyperparameter selection

In this section, we review our findings regarding the observed hyperparameters selections for our experiments. The hyperparameters have been selected by employing nested cross-validation, described in [Section 5.5](#). The best model is determined based on the model hyperparameter configuration with the lowest loss across for the inner cross-validation folds. This results in $K - 1$ experimental configurations, from which the most common hyperparameter choices (mode) used to re-train the model on all inner folds (i.e. no development set). This final model is evaluated on the respective outer fold. This is repeated for all outer folds. Finally, the K outer fold scores are averaged to obtain the overall performance.

7.5.1. Feature extraction parameters

In the following section, we discuss the spectral extraction feature selection, using a LR classifier for the task of binary call activity detection. For this architecture, we tested a broad range of STFT, mel-spectrum and MFCC configurations.

Initially, we experiment by using a single MFCC feature as input to the LR classifier, we denote this model as LR-frame. Our findings indicate that, when only a single MFCC frame is taken as input, increasing the STFT frame length from 25 ms (typical in speech applications) to 100 ms consistently enhanced performance. Although there is a discernible pattern favouring a minimal number of mel triangular filters (64 mel filters), the quantity of MFCC retained does not appear to negatively impact performance. These higher order MFCCs are known to encode speaker pitch information when applied to speech signals. Further subsequent analysis of these coefficients revealed reduced statistical variance of these coefficients compared to the lower order MFCCs. We thus believe that while the use of these higher order cepstral coefficients did not strongly impact classifier performance, their utility when processing elephant vocalisations still remains uncertain.

Subsequently, we considered using multiple neighbouring MFCC features (context window) as input to the classifier and denote this classifier simply as LR in our results. The features from the context window are optionally (hyperparameter) averaged to a single frame, or concatenated together to form a single higher dimensional feature vector, or a dimensionality reduction technique (such as PCA) is applied. In contrast to LR-frame, our experiments show that, when using a context window, reducing the STFT frame length to 50 ms led to improvements for both datasets. The choice of a smaller number of mel filters did not generalise to the context window input experiments. Instead, we found that a larger number of mel filters led to better performance. Furthermore, retaining only the lower order (31) MFCCs lead to further improvements.

As noted before, the higher order MFCCs exhibited reduced variance compared to their lower order counterparts. Further experimentation employed PCA as a dimensionality reduction instead of discarding the higher order MFCCs. The number of principal components to retain was chosen such that the explained variance is 95 of the total features (7 components). The choice of PCA dimensionality reduction technique stemmed for the observed reduced variance in the higher order MFCCs. Employing PCA as a preprocessing step outperformed the naïve approach of merely discarding the higher order coefficients.

The multi-frame LR models exhibited substantially better performance than their single frame counterparts.

7.5.2. *Multi-layer perceptron*

From our experiments, we found that, over both datasets, the MLP architecture and the layerwise dropout probability lead to the most substantial performance enhancement on the development sets. A choice ranging between 10% and 15%, lead to consistent performance improvement. Contrary to expectations, an encoder structure with decreasing hidden layer dimensions as network depth increased did not result in the best performance. Instead, maintaining a consistent hidden layer dimension proved to be the most effective approach.

While deeper MLPs (more than five layers) showed improved performance on certain development folds for the EV dataset, shallower structures, comprising two to four layers (less than 1 million parameters) consistently outperformed the deeper MLP models. However, this trend did not generalise to the LDC dataset, where deeper models consisting of five to seven layers (containing up to 10 million parameters) exhibited superior performance. This can be attributed to the difference in dataset sizes.

The activation function used in the hidden layers had minimal impact on overall performance. Furthermore, batch normalisation adversely affected model performance for both datasets.

7.5.3. *Support vector machine*

In contrast to LR and MLP, no noticeable performance improvement was achieved by optimising the context window, or MFCC configuration when using an SVM classifier. The range of MFCC configurations was not as extensive as for LR-frame, only evaluating the top performing configurations. However, the choice of a long context window with a PCA dimensionality reduction step to five components (92.5% explained variance) consistently leads to better performance. The choice of kernel approximation function

had the most significant impact on overall performance. The Nyström method lead to a reduction of 50 on the developmental loss, compared to the *Random Kitchen Sinks* method, matching the exact kernel performance on the EV dataset. The radial basis function (RBF) was the best overall performing basis function.

7.5.4. ResNet

Out of the standard ResNet configurations (*ResNet-18*, *ResNet-34*, *ResNet-50*, *ResNet-101* and *ResNet-152*) evaluated, *resnet-101* performed the best when evaluated on the LDC dataset. While, the best configuration for the EV dataset was *ResNet-18*. In both cases, a context window of 2.5 s, performed the best overall. There was no clear trend on the choice of learning rate or batch size for training the ResNet models.

7.5.5. AlexNet

As expected, CNN see improved performance with longer context windows. When using AlexNet, the best performance was observed with a context window of 2.5 s, the same as seen for the ResNet architecture.

7.5.6. In-domain pre-training

The AST model was evaluated using model weights that had been pre-trained on AudioSet as well as weights that had a random initialisation. When training from the random initialisation, the model would not generalise well and resulted in poor development set performance.

7.5.7. Out-of-domain pretraining

The impact of out-of-domain pretraining was assessed for both ResNet and AlexNet architectures. It was observed that pretraining these models on out-of-domain ImageNet data resulted in marginal improvements in performance compared to random initialisation. However, when fine-tuning from a pre-trained model, the AST model outperformed all other architectures. Utilising a pre-trained model weights required lowering the initial learning rate of the optimiser. In the case of ResNet, this approach facilitated the use of larger models. However, these did not yield any further enhancements in classification performance. Consequently, employing a smaller ResNet model may be more advantageous due to its better computational efficiency. Pre-training AlexNet on ImageNet led to marginal performance improvements. Remarkably, the optimal model parameters were obtained after just one epoch of training when initialised with pre-trained weights, thereby significantly reducing computation time. Notably, in both ResNet and AlexNet models, a reduction in the initial learning rate by two orders of magnitude was necessary, compared to training from random initialisation, to achieve generalisable performance.

8. Results

The following section described the results obtained for the experiments set out in [Section 7](#).

8.1. Call detection and endpointing

The following experiments assess how well a classification model can perform elephant call activity detection and implicit call endpointing, using the procedures described in [Section 7.1](#). Model detection performance will be assessed based on the AP score, while the endpointing performance will be assessed using the Jaccard index. [Table 4](#) presents experimental results for elephant call detection. The table shows that, across both datasets, the AST-seq achieves better performance than all other considered models, followed by CNN-based models (AlexNet And ResNet) and then the MLP.

The AST-based models clearly outperform the other models, obtaining an AP score of 0.962 for call detection. We speculate this is due to the attention-mechanism used by the transformer-encoder architecture, which allows the model to selectively focus on areas of interest within the extended context window. Other models have a reduced context window length and also do not have the ability to selectively ignore potential adversarial acoustic events. While the AST-seq model makes more errors per time instant than the AST-cls model, informal inspection of the model decisions revealed that these errors tend to occur near the boundaries of the vocalisation segment. Further inspection revealed that some call boundaries postulated by the model may be more precisely aligned with the vocalisation that the human annotated labels do. In contrast, the detection errors made by the other models (in particular the shallow models) are the result of entire vocalisation segments being falsely classified.

From the Jaccard index, we see that the AST-seq model produces the best segment alignment with the ground truth annotations, achieving a Jaccard index of 0.830. However, the high purity and lower coverage indicate that the model tends to over-segment a given audio sequence. Anecdotally, we observe this behaviour particularly in vocalisations that have long tails. In such cases, the model may divide the vocalisation into two or more segments. In contrast, AST-cls tends to under-segment, resulting in missing call boundaries. We observe this in the lower specificity score, indicative of falsely classifying a segment as a vocalisation. When computing the average model output over the segment, inserted boundaries have a smaller impact on the final classification than missing boundaries, thus resulting in improved call detection results. Thus, the AST-seq model is the best performing endpointing model, followed by AST-cls and resnet models, achieving a Jaccard index of 0.798 and 0.788, respectively.

We observe that the convolutional models, AlexNet and ResNet, are also strong contenders for call detection, matching the transformer models in some metrics. We speculate that is due to the positional invariant and therefore robustness of the CNN features. The CNN models also benefit from increased spectral-temporal resolution in comparison to the AST architecture. The first step in the AST architecture is to divide the input spectrogram into patch embeddings, which is a downsampling step. This step is required to improve the computational efficiency of the model. However, it may result in loss of information that is useful in discerning small variations between features. Finally, we see that there is a larger performance gap between the CNN and transformer architecture when more data is available, as is the case with the LDC dataset.

Overall, the deep neural architectures out-performed the shallow classification models in the task of elephant call activity detection. An interesting observation is how well the SVM classifier performs. The performance improvement between SVM and a model of

similar computational complexity such as LR can be attributed to the non-linearity present in the kernel function of the SVM classifier. A notable architectural observation is that all shallow models saw increased performance when provided with an input context of roughly a second in length and the inclusion of a dimensionality reduction step. We speculate that this ensures that the models classifications are more robust to the noisy features extracted from each frame. We did attempt simply averaging the features over time. However, the dimensionality reduction outperformed this approach, particularly after statistical whitening of the reduced numerical feature space.

Figure 10 compares the purity-coverage and ROC curves for a selection of the models presented in Table 4. From the EV purity-coverage curves, we observe that for 95 coverage (i.e. successfully identify 95 of the true positives), LR and Alexnet have an almost equal purity (which of the detections are indeed true positive) of approximately 81. This makes LR a computationally efficient way to perform binary detection for

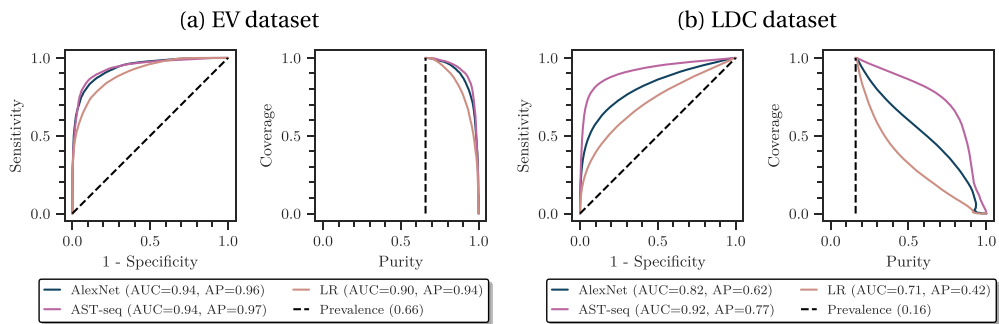


Figure 10. Receiver operating characteristic and coverage-purity curves, for the task of elephant call detection, for both Elephant Voices (EV) and Linguistic Data Consortium (LDC) datasets. The baseline model (LR) and the top performing model (ast-seq) are shown. The prevalence, which is the performance achieved when making random decisions according to the prior.

Table 4. Elephant binary call activity detection results for the elephant voices (EV) and the linguistic data consortium (LDC) datasets for all considered models. The reported metrics are averages over the K outer folds. The associated standard deviation is given in parentheses.

Dataset	Model	Purity	Coverage	AUC ROC	AP	Jaccard
EV ($K = 5$)	LR	0.829 (0.04)	0.798 (0.05)	0.902 (0.02)	0.937 (0.01)	0.719 (0.04)
	SVM	0.853 (0.05)	0.873 (0.04)	0.893 (0.02)	0.928 (0.02)	0.757 (0.03)
	XGB	0.857 (0.05)	0.893 (0.04)	0.907 (0.02)	0.946 (0.02)	0.776 (0.04)
	MLP	0.869 (0.07)	0.860 (0.07)	0.914 (0.03)	0.943 (0.03)	0.754 (0.03)
	AlexNet	0.873 (0.02)	0.891 (0.06)	0.938 (0.02)	0.960 (0.01)	0.787 (0.04)
	ResNet	0.876 (0.04)	0.888 (0.02)	0.933 (0.02)	0.957 (0.01)	0.788 (0.03)
	AST-cl	0.829 (0.03)	0.955 (0.02)	0.938 (0.02)	0.958 (0.01)	0.798 (0.03)
	AST-seq	0.914 (0.04)	0.901 (0.02)	0.940 (0.02)	0.968 (0.01)	0.830 (0.02)
LDC ($K = 10$)	LR	0.286 (0.03)	0.602 (0.04)	0.711 (0.03)	0.419 (0.04)	0.240 (0.02)
	SVM	0.634 (0.08)	0.178 (0.02)	0.727 (0.02)	0.404 (0.05)	0.161 (0.01)
	XGB	0.664 (0.07)	0.130 (0.02)	0.694 (0.02)	0.370 (0.05)	0.121 (0.02)
	MLP	0.665 (0.03)	0.241 (0.02)	0.684 (0.03)	0.421 (0.03)	0.201 (0.04)
	AlexNet	0.691 (0.07)	0.498 (0.03)	0.825 (0.03)	0.616 (0.04)	0.403 (0.03)
	ResNet	0.633 (0.04)	0.490 (0.03)	0.812 (0.03)	0.606 (0.04)	0.382 (0.05)
	AST-cl	0.768 (0.05)	0.578 (0.05)	0.886 (0.04)	0.716 (0.04)	0.504 (0.04)
	AST-seq	0.768 (0.05)	0.700 (0.04)	0.918 (0.02)	0.772 (0.06)	0.578 (0.05)

further downstream tasks. From the ROC curves, we see that the AST-seq model almost always exhibits better performances. Furthermore, we observe from the prevalence (the ratio of positive samples to the total number of samples) that the EV dataset is more balanced than the LDC dataset. Comparing the ROC and purity–coverage curves, we see that the performance of both classifiers has decreased for the LDC dataset.

As discussed in [Section 6](#), elephant call detection can also be viewed as a multi-label task, where not just the presence of a call is detected, but in the same step also the type of call. [Figure 11](#) compares the purity–coverage curves for an LR, AlexNet and AST-seq models in this multi-label scenario, evaluated on a context window basis. We observe that while the LR performs well for rumble detection, the AST-seq model offers vastly superior performance for the roar and trumpet classes. Furthermore, we note that none of the classifiers could correctly detect more than 50% of the cry segments, reflected by the coverage score. This may be due to the short duration and high energy present in the call type. As a result of the acoustic properties of the call, there is no harmonic or formant structure to the call type. The poor performance may also be attributed to the small number of samples available ([Figure 2](#)), where there are roughly only two exemplars per cross-validation fold.

The sudden jump in purity is the result of the classifier correctly identifying a large portion of the classification segment after a small change in the classification threshold. We found that for these exemplars, the spectral features vectors showed a high degree of correlation (cosine similarity) between frames. As a result, we expect these features to be close in the respective classifier feature space.

From both [Table 5](#) and [Figure 11](#), we observe that while the shallow classifiers achieved commendable performance in the task of binary detection, these models perform far worse than the deeper architectures for the multi-label tasks. The AST-based models, however, remain the top performing candidates.

8.2. Call classification

Next, we assess the classifier’s performance on a segment-level elephant call classification task in a multi-label scenario. As stated before, it is assumed that this task is downstream from an oracle endpointing model, and thus the segment location in time is computed from the annotation labels. Thus, this task assesses how well a model is able to classify an entire segment of audio.

We observe that the models that perform well at call activity detection (including negative class), are also the models that perform well at segment-level classification (excluding negative class) tasks, seen by comparing the results in [Tables 5](#) and [6](#).

Furthermore, we observe the same trend in segment-level classification of low-resource classes, such as cry, that we saw for detection tasks. [Figure 12](#), shows how the worst performing class remains the cry-class. However, in the case of the segment-level classification, the AST and other deep architectures all overfit (memorised training examples) on the cry-class, while the shallow classifiers showed superior performance. In contrast, the opposite is true when evaluating the models in a multi-label detection scenario. Such overfitting could in future be addressed by employing low-resource techniques such as few-shot learning, where

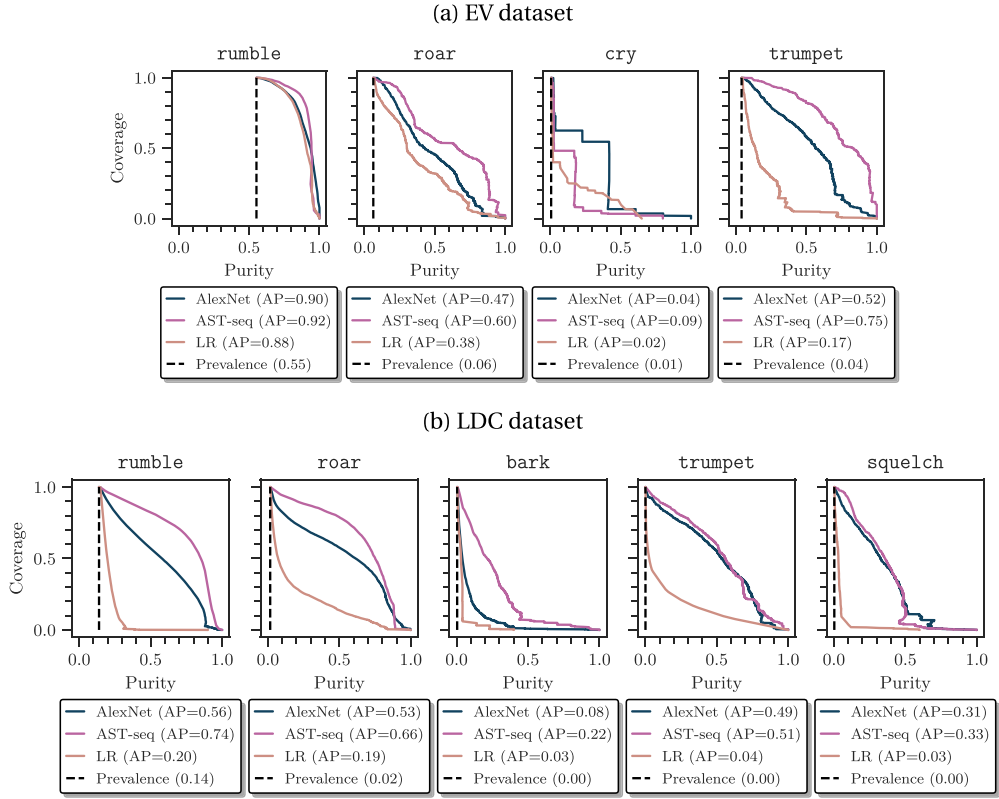


Figure 11. Purity-detection, both Elephant Voices (EV) and linguistic data consortium (LDC) datasets. The baseline coverage curves, for the task of multi-label elephant call activity model (LR) and the top performing model (AST-seq) are shown. The prevalence, which is the performance achieved when making random decisions according to the prior.

Table 5. Multi-label elephant call detection results for the elephant voices (EV) and the linguistic data consortium (LDC) datasets for all considered models. The reported metrics are averages over the K outer folds. The associated standard deviation is given in parentheses.

Dataset	Model	Purity	Coverage	AUC ROC	map	Jaccard
EV ($K = 5$)	LR	0.299 (0.04)	0.511 (0.16)	0.798 (0.09)	0.370 (0.08)	0.240 (0.04)
	SVM	0.433 (0.09)	0.310 (0.06)	0.839 (0.08)	0.379 (0.06)	0.249 (0.04)
	XGB	0.507 (0.07)	0.279 (0.04)	0.833 (0.08)	0.406 (0.05)	0.240 (0.03)
	MLP	0.397 (0.12)	0.318 (0.12)	0.853 (0.07)	0.416 (0.07)	0.258 (0.08)
	AlexNet	0.515 (0.10)	0.385 (0.10)	0.895 (0.07)	0.484 (0.09)	0.314 (0.08)
	ResNet	0.407 (0.13)	0.259 (0.05)	0.847 (0.08)	0.385 (0.08)	0.217 (0.05)
	AST-cls	0.530 (0.10)	0.477 (0.12)	0.871 (0.09)	0.514 (0.09)	0.374 (0.09)
LDC ($K = 10$)	AST-seq	0.597 (0.06)	0.475 (0.10)	0.870 (0.09)	0.591 (0.10)	0.409 (0.07)
	LR	0.048 (0.01)	0.628 (0.16)	0.722 (0.10)	0.010 (0.06)	0.043 (0.01)
	SVM	0.161 (0.07)	0.018 (0.01)	0.711 (0.08)	0.082 (0.03)	0.017 (0.01)
	XGB	0.520 (0.14)	0.068 (0.03)	0.776 (0.07)	0.217 (0.09)	0.065 (0.03)
	MLP	0.461 (0.20)	0.114 (0.05)	0.825 (0.06)	0.254 (0.10)	0.105 (0.05)
	AlexNet	0.579 (0.19)	0.301 (0.10)	0.888 (0.04)	0.390 (0.11)	0.248 (0.08)
	ResNet	0.523 (0.19)	0.276 (0.10)	0.883 (0.04)	0.358 (0.11)	0.221 (0.07)
	AST-cls	0.543 (0.13)	0.380 (0.10)	0.918 (0.03)	0.434 (0.11)	0.295 (0.07)
	AST-seq	0.586 (0.13)	0.411 (0.12)	0.952 (0.03)	0.491 (0.12)	0.328 (0.10)

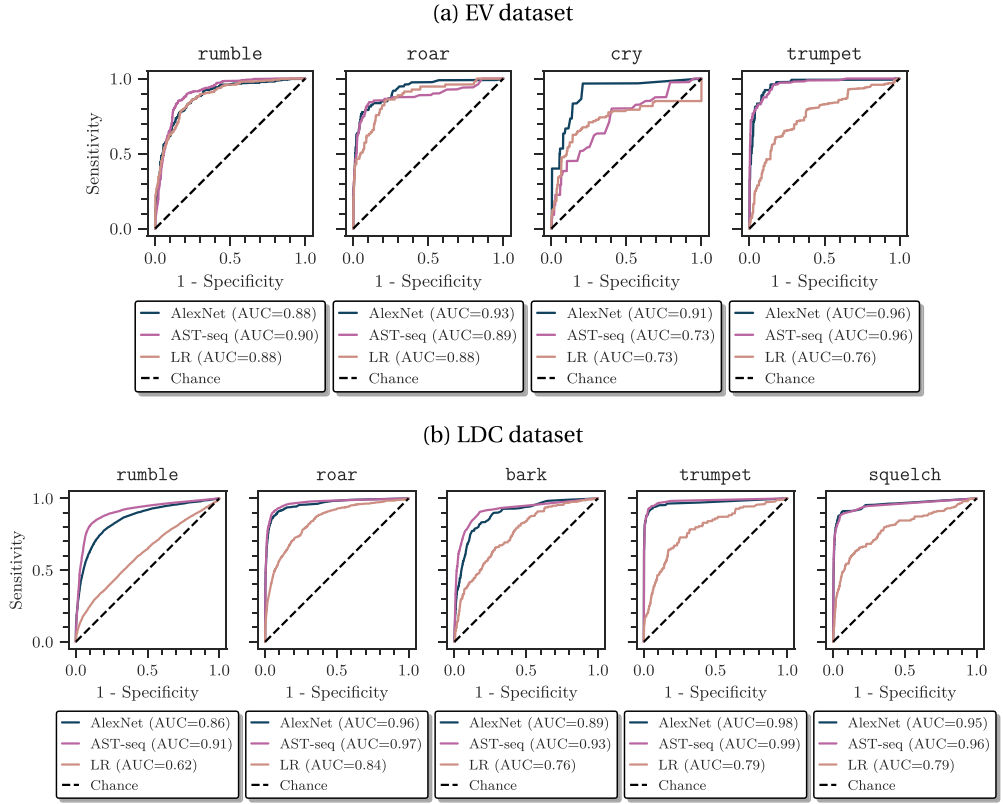


Figure 12. Receiver Operating Characteristic (ROC) curves, for the task of segment-level elephant multi-label subcall classification, for elephant voices (EV) dataset. The baseline model (LR) and the top performing model (AST-seq) are shown. The chance, which is the performance achieved when making random decisions according to the prior.

Table 6. Segment-level elephant multi-label call classification results for the Elephant Voices (EV) and the Linguistic Data Consortium (LDC) datasets for all considered models. The reported metrics are averages over the K outer folds. The associated standard deviation is given in parentheses.

Dataset	Model	Precision	Recall/Sens.	Specificity	AUC ROC
EV ($K = 5$)	LR	0.325 (0.13)	0.525 (0.27)	0.908 (0.06)	0.849 (0.12)
	SVM	0.453 (0.17)	0.257 (0.06)	0.947 (0.03)	0.876 (0.09)
	XGB	0.416 (0.15)	0.245 (0.05)	0.951 (0.02)	0.866 (0.08)
	MLP	0.436 (0.19)	0.304 (0.12)	0.942 (0.02)	0.880 (0.08)
	AlexNet	0.492 (0.23)	0.373 (0.13)	0.944 (0.02)	0.920 (0.05)
	ResNet	0.406 (0.25)	0.269 (0.07)	0.932 (0.05)	0.881 (0.04)
	AST- cls	0.529 (0.10)	0.466 (0.12)	0.954 (0.03)	0.876 (0.12)
	AST-seq	0.533 (0.08)	0.458 (0.12)	0.957 (0.01)	0.871 (0.09)
	LR	0.119 (0.05)	0.640 (0.20)	0.744 (0.11)	0.759 (0.12)
LDC ($K = 10$)	SVM	0.200 (0.18)	0.001 (0.01)	0.999 (0.01)	0.758 (0.10)
	XGB	0.409 (0.17)	0.039 (0.03)	0.999 (0.01)	0.775 (0.07)
	MLP	0.470 (0.25)	0.095 (0.06)	0.996 (0.00)	0.867 (0.07)
	AlexNet	0.638 (0.17)	0.320 (0.15)	0.993 (0.01)	0.929 (0.05)
	ResNet	0.653 (0.26)	0.302 (0.15)	0.991 (0.01)	0.916 (0.05)
	AST-cls	0.690 (0.28)	0.390 (0.13)	0.988 (0.01)	0.954 (0.03)
	AST-seq	0.730 (0.20)	0.435 (0.13)	0.987 (0.01)	0.957 (0.03)

a nearest neighbour approach is used in conjunction on the feature representation obtained from pre-trained neural network.

8.3. Subcall classification

As discussed in [Section 5.4](#), elephants produce a set of nuanced calls that have been shown to strongly correlate with animal behaviour. The ability to automatically identify such subcalls would greatly benefit animal monitoring and provide near-realtime indications on herd health and status.

As before, we employ several shallow and deep classification models in order to perform segment-level classification of these subcalls. In order to maintain the nested cross-validation experimental setup, only a subset of the subcalls can be evaluated ([Figure 3](#)). Furthermore, only the EV dataset can be used since only it contains the required subcall annotations. Hence, the LDC dataset is not used in the subcall experiment.

[Figure 13](#) shows that the ROC curves are associated with the results presented in [Table 7](#). As before, we observe that the shallow models, while strong contenders in the binary call detection task, do not perform well in these multi-label tasks scenarios (including subcall classification). Due to the small number of samples available to train the deeper models, overfitting occurs. However, in contrast to segment-level call classification, where the shallow models performed better on these low resources classes (such as cry) this is not the case for subcall classification. We postulate that subcalls may in fact be a more difficult classification problem, thus while the deeper architectures may overfit, when incorporating a pre-trained network, this risk is reduced. Thus, we ultimately use the deep neural networks as feature extractors (encoder structure) and rely on the final linear layer to perform the actual classification.

As before, we see that the transformer models achieve best performance followed by the CNN-based models, and then MLP.

9. Discussion

This section examines the high-dimensional speech-based features compared against transformer-based representations. Furthermore, we analyse the attention map of the AST model to gain better insight into its internal mechanisms. Finally, we conclude with a broad discussion on the overall trends observed across the different experiments.

9.1. Feature space analysis

The results of our multi-label call activity detection experiments suggest that using speech-based features as input might be limiting the performance of certain models. Although the effectiveness and applicability of these features beyond speech research is contested, these features are considered essential when utilising pre-trained models. To evaluate this hypothesis, we compare the high-dimensional MFCC and mel-spectral features using a uniform manifold approximation and projection (UMAP) for visualisation of the high-dimensional data. We compare these input features along with the pre-trained AST representations.

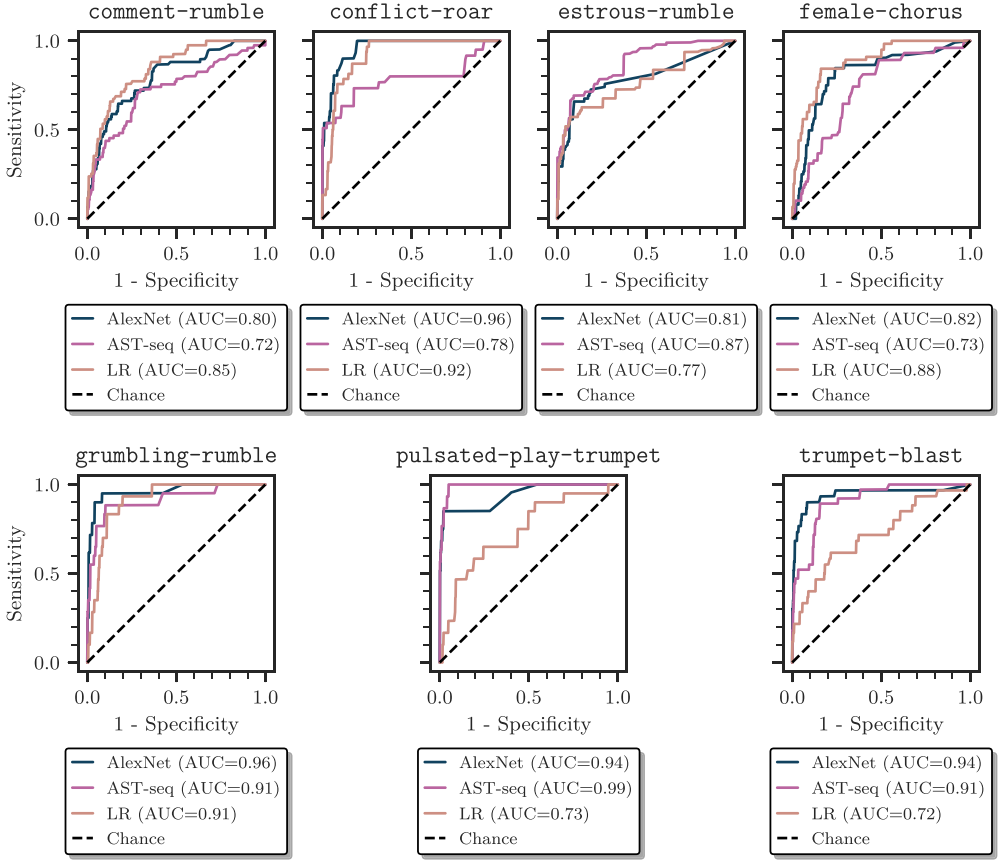


Figure 13. Receiver Operating Characteristic (ROC) curves, for the task of segment-level elephant multi-label subcall classification, for elephant voices (EV) dataset. The baseline model (LR) and the top performing model (ast-seq) are shown. The chance, which is the performance achieved when making random decisions according to the prior.

Table 7. Segment-level elephant multi-label subcall classification results for the Elephant Voices (EV) dataset for all considered models. The reported metrics are averages over the K outer folds. The associated standard deviation is given in parentheses.

Dataset	Model	Precision	Recall/Sens.	Specificity	AUC ROC
EV ($K = 5$)	LR	0.141 (0.08)	0.680 (0.03)	0.824 (0.12)	0.826 (0.10)
	SVM	0.029 (0.06)	0.007 (0.02)	0.992 (0.02)	0.826 (0.14)
	XGB	0.107 (0.15)	0.037 (0.07)	0.999 (0.00)	0.864 (0.12)
	MLP	0.131 (0.20)	0.056 (0.08)	0.999 (0.00)	0.875 (0.12)
	AlexNet	0.395 (0.36)	0.159 (0.14)	0.998 (0.00)	0.888 (0.09)
	ResNet	0.246 (0.30)	0.116 (0.14)	0.998 (0.00)	0.875 (0.09)
	AST-cls	0.613 (0.31)	0.325 (0.20)	0.996 (0.01)	0.979 (0.10)
	AST-seq	0.374 (0.27)	0.251 (0.23)	0.993 (0.01)	0.845 (0.12)

We obtain the AST representations by computing the output of the last layer of the transformer (before the MLP classification layer) for a given input context. As the model produces a sequence of outputs both for the spectral and temporal input dimensions, we

average the outputs associated with the spectral input dimension to obtain a single 768-dimensional vector every 160 ms in time.

Each of the respective high-dimensional features and representations are reduced to a two-dimensional projection using a UMAP (McInnes et al. 2020). The UMAP is calibrated using the training labels and applied to the development set.

Figure 14 shows the two-dimensional UMAP for a single inner cross-validation fold of the EV dataset. MFCC, mel-spectral features, and AST representations were extracted from both training and development fold. It is evident that, while discernible clusters form in the training projections for all three feature types, the spectral features (mel and MFCC) do not generalise well to the development fold. Despite this, a clear separation between the two classes no-call and rumble is observed, for all features. This could explain the high performance of shallow models in binary call detection but poor performance in multi-label call detection. The AST features do, however, generalise well to the validation fold and maintain similar cluster shape.

Upon examination of misclassifications made by the shallow models, it was found that the classifier struggled with highly polyphonic segments. These polyphonic instances typically include noise, such as engine sounds, wind rustling foliage; or other dominant sound sources such as birds, insects, or combinations thereof. Identifying the precise call boundary for such instances proved challenging even for human annotators. We believe that the low-energy tails of such vocalisations were only noticed by the annotators due to the high-energy start of the vocalisation. Additional misclassifications were attributed to recording artefacts, such as microphone handling sounds and unanticipated starting or ending instances within recordings. It is noteworthy that some misclassifications might even be considered legitimate, as certain ground truth labels marginally fall short of the threshold required for a positive sample classification.

Deep-classification models with larger temporal contexts faced similar challenges, although resulting in fewer misclassifications in the presence of polyphonic source environments. However, the CNN models still misclassify low-energy vocalisations with extended tails. The AST-based models were the only classifiers that could accurately detect low-energy vocalisation in a highly polyphonic settings, likely due to the global attention mechanism present in its transformer-encoder architecture.

9.2. Attention map analysis

We now will consider the attention map used by the AST-seq model for a particular exemplar drawn from a development fold of the EV dataset. The transformer self-attention mechanism, as briefly described, allows to model to selectively focus on particular features in the input, based on other input features. This characteristic of the attention mechanism enables contextualised classifications based on past and future features in the extended input context. We postulate that this is the reason for the models improved ability to detect low-energy vocalisations amidst other acoustic sources.

Figure 15 shows the attention map for one isolated model output (token), in a binary call detection scenario. Note that the only model architecture that correctly identified this example segment was AST-seq. From the attention map, we can see that the model uses features from both the past and the future to produce the chosen classification output. The majority of the input features are masked by the attention map, but not for the region

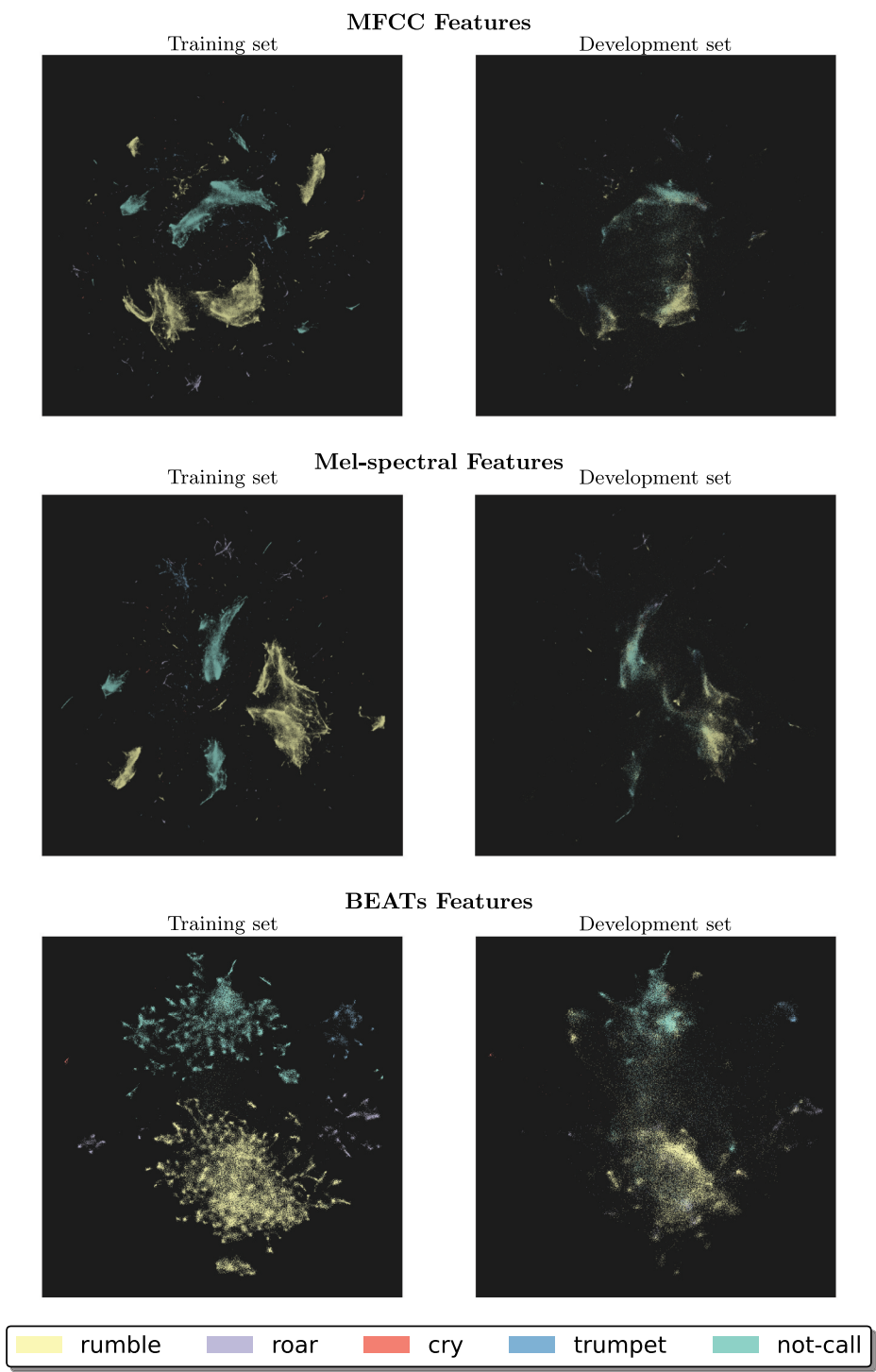


Figure 14. Supervised UMAP of MFCC, mel-spectral and BEATs features for a particular inner cross-validation turn.

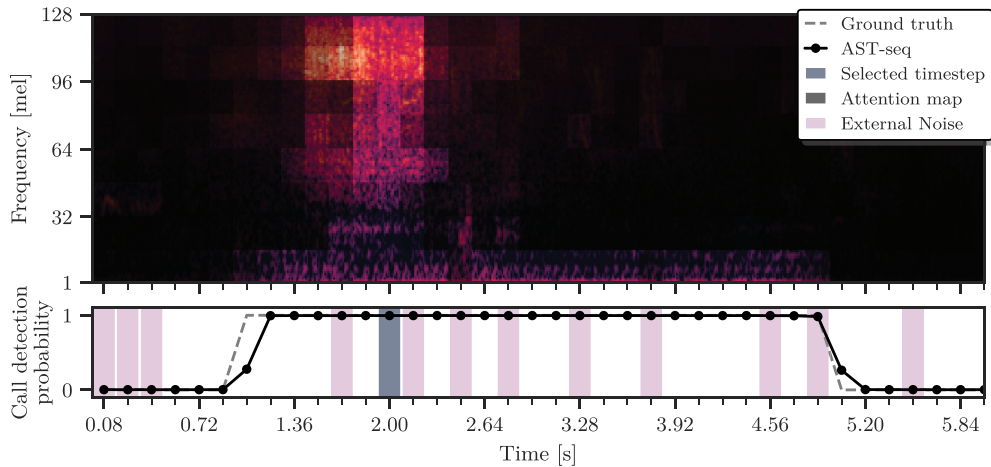


Figure 15. Mel-spectrogram input to ast-seq model with binary call detection probability output shown for time step at 2s. Attention map overlay shown, darker regions of the attention map shows features rejected by the attention mechanism, and vice versa. In addition external noise sources (e.g. bird calls) are also indicated. A detection probability is produced per temporal patch every 160 ms. The exemplar is drawn from the development set of the given cross-validation fold.

corresponding to the low-frequency elephant rumble. We also see that features from other external noise sources are correctly rejected. The focused attention allows the model to better reject acoustic sources that may have an adversarial effect on the model performance.

From Figure 15 we see that the AST-seq model is also attending to acoustic sources other than elephants, e.g. bird calls (indicated as *external noise* in the figure). This could mean that the model is using these external audio sources as a proxy for elephant vocalisation, or that the model is using these calls to reject other external sources. With the limited data available in the EV dataset, it is not yet possible to discern the mechanism at play.

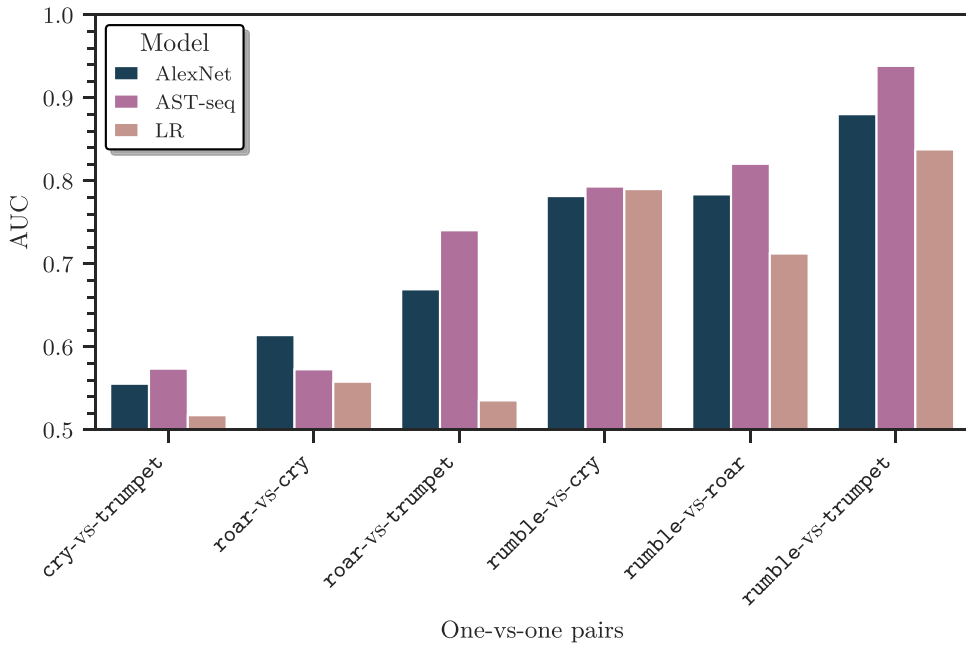
9.3. Inter-call classification analysis

This section discusses the comparative one-vs-one call classification results for LR, AlexNet, and AST-seq models. The one-vs-one classification procedure involves pairing calls and scoring them in a binary fashion to consider how well a classifier can distinguish between two different call types (Section 7.4.1). This differs from the previous analysis that considered how well the classifier can distinguish a particular call from acoustic sources other than elephant calls (not-call). Since such one-vs-one scoring cannot be performed when the two calls in question are present simultaneously, all classification intervals where such overlap occurs have been omitted from the one-vs-one scoring.

Figure 16 shows the one-vs-one multi-label AUC detection scores for both EV and LDC datasets evaluated on the call-level annotation.

In the case of the EV dataset, we observe that the AST-seq achieves the best performance in all cases except when distinguishing between roar and cry call types. As previously observed, minority classes, such as cry and trumpet, are associated with

(a) EV call dataset.



(b) LDC call dataset.

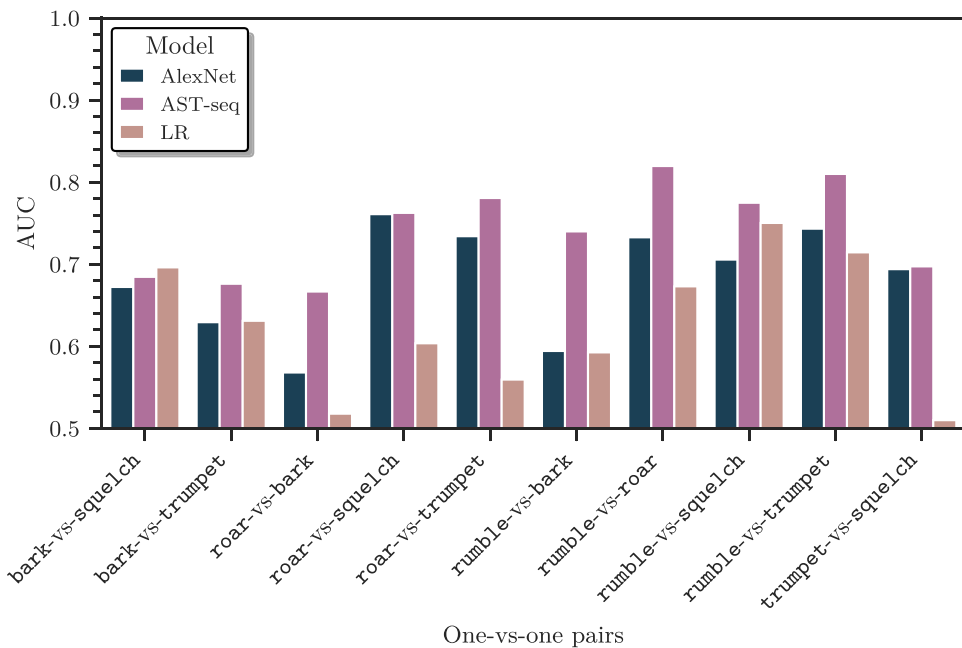


Figure 16. Area Under the Curve (AUC) scores, for the task of multi-label one-vs-one elephant call detection for (a) The elephant voices (EV) and (b) The Linguistic Data Consortium (LDC) dataset. A selection of three models are shown, LR (baseline), AlexNet and ast-seq (top performing). In each case the classification performance (AUC score) between two specific classes, are compared using one positive class against one negative class in a one-vs-one approach. Importantly, the performance score remains identical if the class labels are interchanged – for instance, the performance of cry-vs-rumble is equivalent to rumble-vs-cry.

poor performance. This is particularly noticeable in deep classification models like alexnet and AST-seq, which typically require more data to generalise well. The good performance of the LR classifier in distinguishing between rumble and cry call types suggests that this task is largely linearly separable. This is not surprising as the acoustic properties of these two calls are quite different, with cry containing predominantly high-frequency information, while rumbles are known for their low-frequency content.

For the LDC (Figure 16 (b)) dataset we observe a similar trend with AST-seq exhibiting improved AUC scores across all call pairs when compared to both AlexNet and LR. Although both AlexNet and AST-seq models perform equally well in distinguishing between trumpet and squelch call types, LR performs on par with a random classifier, achieving only an AUC score of 0.51. Likewise, both AlexNet and LR perform poorly in differentiating roar and bark call types, while the AST-seq model performs acceptably well.

We conclude that the AST-seq model not only excels at distinguishing elephant calls from other environmental sounds but also that the model is able to distinguish between different elephant calls.

9.4. Classifier performance

Here, we broadly discuss the overall trends observed by classifier performance across both datasets and the various tasks reported in Section 8.

First, we compare LR and SVM. These models have similar computation complexity and both rely on a hyperplane decision boundary. Due to the non-linear kernel function of the SVM model, the decision boundary is no longer linear as is the case with LR. As a result, we observe that the SVM model outperforms the LR models in the majority of cases. However, this additional kernel function comes at an increased computational and memory complexity. Furthermore, if an output probability is required, the SVM classifier requires an additional classifier calibration step. Thus, as a lightweight model for early preprocessing of binary call detection (for later downstream tasks), LR is a model that has computational cost to performance balance.

XGB had similar performance to the MLP models. However, the hyperparameter search was far more computationally expensive to perform for MLP. This is in part due to the large hyperparameter space that the MLP models occupy and how increases in the depth or width of the model increase its computational complexity. In contrast, the XGB model hyperparameters had less variance over the search, indicating that the model was more robust to naïve model parameter selection. Furthermore, many of the XGB model parameters result in a reduction in the model capacity, by reducing for instance the number of leaf nodes or trees. As such, searching the hyperparameter space generally leads to computational improvements.

When considering the CNN-based models, we observe that the overall alexnet either exceeded or matched the resnet performance. This is counter intuitive, as the residual connections in resnets allow for deeper models to be trained. However, due to the small size of the datasets used in this work, we do not observe these benefits to performance in our experimental results.

Finally, we compare the two best performing models: AST-cls and AST-seq. The only difference between these transformer-based models is in their output layer. However, we observe in some cases that there is a noticeable performance difference between these two

models. For most endpointing metrics, the AST-seq model is superior. We postulate that this is due to the sequence-to-sequence training regime, where the explicit call segment boundaries are presented to the model during training, while for the AST-cls model this has to be implicitly learned as it is only given a single output target. However, this does also make detections near the start and the end of the input context more difficult for the AST-seq model. While the AST-cls model always produces a model classification of the centre of the input context, the AST-seq model is tasked with producing classification outputs near the start and the end of the same input context, with less local contextual information. Hence, AST-seq is better at call endpointing, while AST-cls is better at detection and classification.

10. Summary and conclusion

We have considered the automatic detection, endpointing and classification of elephant vocalisations in audio recordings. A combination of both shallow and deep architectures were evaluated against two audio datasets containing elephant vocalisations. These datasets consist of multi-label classifications targets. By performing call activity detection using a classification model applied to a series of context windows over time, we are able to compute the multi-label call probability every 100 ms. From these discrete context window detections, we are able to localise the call in time and endpoint at the start and the end of the call in the recording. Furthermore, we evaluate the model's ability to identify an unknown elephant vocalisation from an endpointed audio segment.

We have evaluated LR, SVM and XGB as our chosen shallow model architectures, and MLP, CNN-based AlexNet and ResNet architectures and finally the transformer-based AST models as the chosen deep model architectures. These models have been evaluated on a large range of hyperparameters, using nested cross-validation, in an attempt to find the best selection of model parameters for the given datasets.

We have also investigated the use of a dimensionality reduction pre-processing step and MFCC and mel-spectral feature configurations. For shallow models, an input context length of 500 ms to 1000 ms performed best, followed by a PCA dimensionality reduction step. For the CNN-based models, an input length of 2.5 s lead to the best performance.

For the deep architectures, transfer learning using out-of-domain pre-trained networks was found to lower the overall training time but lead to only minor or no performance improvements compared to starting from random initialisation. In contrast, using in-domain pre-trained models lead to both a reduction in training time and improved model performance. The application of task transfer by finetuning a model pre-trained to produce a single classification output to produce a sequence of classification outputs drastically improves the computational efficiency of the model and also leads to the best performing model (AST-seq).

The AST-based models performed the best in all three tasks: call detection (both binary and multi-label), call endpointing and call classification. However, we also find that LR is a viable lightweight model for binary call detection. In both multi-label call detection and classification tasks, it was found that the shallow classification models lead to a deterioration in performance, compared to binary call detection. We speculate that this is because the MFCC input feature space used by the shallow models not representative of the call characteristics.

In conclusion, this study has demonstrated the significant benefits of pre-training and ensuing success of sophisticated models, in the context of small bioacoustic datasets. The AST-seq model emerged as a top performer in terms of both computational efficiency, due to the sequence-to-sequence structure, and detection as well as classification performance. Additionally, we found that, while speech features perform well in detection tasks, they are not as effective for classification purposes. Finally, transformer models offer a promising avenue for sub-call classification, demonstrating their successful application in early automated animal behavioural classification.

Notes

1. AudioSet is a large-scale dataset consisting of over 2 million 30-second sound clips, each labelled with one or more of 527 audio event classes. It has been widely used in audio classification and self-supervision tasks. The dataset covers a wide range of categories, from music and speech to nature sounds and human activities.
2. This differs from Zeppelzauer et al. (2013), who used the maximum classifier output probability instead.
3. The extended context window is typically an order of magnitude longer than the typical context window.

Acknowledgements

We gratefully acknowledge financial support by Telkom (South Africa) for the research presented in this paper. We would also like to thank the Stellenbosch Rhasatsha High Performance Computing facility and team for access to their facilities and technical support. Their expertise and resources were invaluable to the successful completion of this project. We thank the contributors of the open-source software packages *scikit-learn* (Pedregosa et al. 2011), *PyTorch* (Paszke et al. 2019) and *Ray* (Moritz et al. 2018) for developing these key tools. Finally, we would like to acknowledge NVIDIA for their generous donation of GPU which provided additional computational resources, which were key to the training and evaluation of the neural network classifiers.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Christiaan M. Geldenhuys  <http://orcid.org/0000-0003-0691-0235>

Thomas R. Niesler  <http://orcid.org/0000-0002-7341-1017>

References

- Bishop CM, Nasrabadi NM. 2006. Pattern recognition and machine learning. Vol. 4. New York, NY, USA: Springer.
- Bjorck J, Rappazzo BH, Chen D, Bernstein R, PH Wrege, Gomes CP. 2019. Automatic detection and compression for passive acoustic monitoring of the African forest elephant. Proceedings of the AAAI conference on artificial intelligence; Honolulu, (HI), USA. p. 476–484. [10.1609/aaai.v33i01.3301476](https://doi.org/10.1609/aaai.v33i01.3301476).

- Breiman L. 2001. Random forests. *Mach Learn.* 45(1):5–32. doi: [10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324).
- Chen S, Wu Y, Wang C, Liu S, Tompkins D, Chen Z, Wei F. 2022. Beats: audio pre-training with acoustic tokenizers. *arXiv preprint arXiv:2212.09058*.
- Chen T, Guestrin C. 2016. Xgboost: a scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. arXiv:1603.02754 [cs]. San Francisco (USA); 785–794. doi: [10.1145/2939672.2939785](https://doi.org/10.1145/2939672.2939785).
- Clemins PJ, Johnson MT. 2003. Application of speech recognition to African elephant (*Loxodonta africana*) vocalizations. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*; China: Hong Kong Vol. 1. p. 1–I. doi: [10.1109/ICASSP.2003.1198823](https://doi.org/10.1109/ICASSP.2003.1198823).
- Clemins PJ, Johnson MT, Leong KM, Savage A. 2005. Automatic classification and speaker identification of African elephant (*Loxodonta africana*) vocalizations. *The J Acoust. Soc. Am.* 117(2):956–963. doi: [10.1121/1.1847850](https://doi.org/10.1121/1.1847850).
- Cortes C, Vapnik V. 1995. Support-vector networks. *Mach Learn.* 20(3):273–297. doi: [10.1007/BF00994018](https://doi.org/10.1007/BF00994018).
- Deng J, Dong W, Socher R, LJ Li, Li K, Fei-Fei L. 2009. ImageNet: a large-scale hierarchical image database. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. ISSN: 1063-6919; Miami (USA); 248–255. doi: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848).
- de Silva S. 2010a. Acoustic communication in the Asian elephant, *Elephas maximus*. *Behaviour.* 147(7):825–852. doi: [10.1163/000579510X495762](https://doi.org/10.1163/000579510X495762).
- de Silva S. 2010b. Aug. Asian elephant vocalisations. [10.35111/7sj22437](https://doi.org/10.35111/7sj22437).
- Devlin J. 2019. BERT: pre-training of deep bidirectional transformers for language understanding by Burstein J, Doran C Solorio T editors. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*; Minneapolis (USA): Association for Computational Linguistics. p. 4171–4186. doi: [10.18653/v1/n19-1423](https://doi.org/10.18653/v1/n19-1423).
- Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al. 2020. An image is worth 16x16 words: transformers for image recognition at scale. *Proceedings of the International Conference on Learning Representations*; Austria. doi: [10.48550/arXiv.2010.11929](https://doi.org/10.48550/arXiv.2010.11929).
- Fukushima K. 1980. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol Cybern.* 36(4):193–202. doi: [10.1007/BF00344251](https://doi.org/10.1007/BF00344251).
- Geldenhuys CM. 2023. Kleinvoet: a spatially-distributed temporally synchronised infrasonic recorder [MEng thesis]. Stellenbosch: University. [10.10019.1/128931](https://doi.org/10.10019.1/128931).
- Gemmeke JF, DP Ellis, Freedman D, Jansen A, Lawrence W, RC Moore, Plakal M, Ritter M. 2017. Audio set: an ontology and human-labeled dataset for audio events. *Proceedings of IEEE international conference on acoustics, speech and signal processing (ICASSP)*; New Orleans, (LA), USA: IEEE. p. 776–780. doi: [10.1109/ICASSP.2017.7952261](https://doi.org/10.1109/ICASSP.2017.7952261).
- Gobush KS, Edwards CTT, Balfour D, Wittemyer G, Maisels F, Taylor RD. 2020. *Loxodonta africana* (amended version of 2021 assessment). The IUCN Red List of Threatened Species. doi: [10.2305/IUCN.UK.2022-2.RLTS.T181008073A223031019.en](https://doi.org/10.2305/IUCN.UK.2022-2.RLTS.T181008073A223031019.en).
- Gobush KS, Edwards CTT, Maisels F, Wittemyer G, Balfour D, Taylor RD. 2020. *Loxodonta cyclotis* (errata version published in 2021). The IUCN Red List of Threatened Species. doi: [10.2305/IUCN.UK.2021-1.RLTS.T181007989A204404464.en](https://doi.org/10.2305/IUCN.UK.2021-1.RLTS.T181007989A204404464.en).
- Gong Y, Chung Y-A, Glass J. 2021. AST: audio spectrogram transformer. *Proceedings of Interspeech*. arXiv:2104.01778 [cs]. Brno, Czech; 571–575. doi: [10.21437/Interspeech.2021-698](https://doi.org/10.21437/Interspeech.2021-698).
- He K. 2016. Deep residual learning for image recognition. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*; Las Vegas, (NV), USA; 770–778. doi: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).
- Hershey S, Chaudhuri S, DP Ellis, JF Gemmeke, Jansen A, RC Moore, Plakal M, Platt D, RA Saurous, Seybold B, Slaney M. 2017. CNN architectures for large-scale audio classification. In *Proceedings of IEEE international conference on acoustics, speech and signal processing (ICASSP)*. arXiv:1609.09430 [cs, stat]. New Orleans, (LA), USA; 131–135. doi: [10.1109/ICASSP.2017.7952132](https://doi.org/10.1109/ICASSP.2017.7952132).

- Hu H. 2018. Relation networks for object detection. In *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*; Salt Lake City, (UT), USA: Computer Vision Foundation IEEE Computer Society. p. 3588–3597. doi: [10.1109/CVPR.2018.00378](https://doi.org/10.1109/CVPR.2018.00378).
- Keen SC, Shiu Y, Wrege PH, Rowland ED. 2017. Automated detection of low-frequency rumbles of forest elephants: a critical tool for their conservation. *The J Acoustical Soc Am*. 141 (4):2715–2726. doi: [10.1121/1.4979476](https://doi.org/10.1121/1.4979476).
- Kemp T, Schmidt M, Westphal M, Waibel A. 2000. Strategies for automatic segmentation of audio data. *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*; Istanbul, Turkey, Vol. 3. p. 1423–1426. doi: [10.1109/ICASSP.2000.861862](https://doi.org/10.1109/ICASSP.2000.861862).
- Kingma DP, Ba J. 2014. Adam: a method for stochastic optimization. *CoRR*.
- Krizhevsky A, Sutskever I, Hinton GE. 2012. ImageNet classification with deep convolutional neural networks. *Proceedings of Advances in Neural Information Processing Systems (NIPS)*; Lake Tahoe, (NV), USA: Curran Associates, Inc. Vol. 25.
- Langbauer WR Jr. 2000. Elephant communication. *Zoo Biol*. 19(5):425–445. doi: [10.1002/1098-2361\(2000\)19:5<425::AID-ZOO11>3.3.CO;2-1](https://doi.org/10.1002/1098-2361(2000)19:5<425::AID-ZOO11>3.3.CO;2-1).
- LeCun Y, Boser B, Denker J, Henderson D, Howard R, Hubbard W, Jackel L. 1989. Handwritten digit recognition with a back-propagation network. *Adv Neural Inf Process Syst*. 2:2.
- Leong KM. 2003. Quantifying acoustic and temporal characteristics of vocalizations for a group of captive African elephants *Loxodonta africana*. *Bioacoustics*. 13(3):213–231. doi: [10.1080/09524622.2003.9753499](https://doi.org/10.1080/09524622.2003.9753499).
- Leonid T, Jayaparthathy R. 2022. Classification of elephant sounds using parallel convolutional neural network. *Intell Autom Soft Comput*. 32(3):1415–1426. doi: [10.32604/iasc.2022.021939](https://doi.org/10.32604/iasc.2022.021939).
- Leung T, Malik J. 2001. Representing and recognizing the visual appearance of materials using three-dimensional textons. *Int J Comput Vision*. 43(1):29–44. doi: [10.1023/A:1011126920638](https://doi.org/10.1023/A:1011126920638).
- Liu S, Deng W. 2015. Very deep convolutional neural network based image classification using small training sample size. In *Proceedings of IAPR Asian Conference on Pattern Recognition (ACPR)*; Kuala Lumpur, Malaysia; 730–734. doi: [10.1109/ACPR.2015.7486599](https://doi.org/10.1109/ACPR.2015.7486599).
- Lorena AC, De Carvalho ACPLF, Gama JMP. 2008. “A review on the combination of binary classifiers in multiclass problems”. en. *Artif Intell Rev*. 30(1–4):19–37. doi: [10.1007/s10462-009-9114-9](https://doi.org/10.1007/s10462-009-9114-9).
- McInnes L, Healy J, Melville J. 2020. UMAP: uniform manifold approximation and projection for dimension reduction. [10.48550/arXiv.1802.03426](https://arxiv.org/abs/1802.03426).
- McInnes L, Healy J, Saul N, Großberger L. 2018. UMAP: uniform manifold approximation and projection. *J Open Source Softw*. 3(29):861. doi: [10.21105/joss.00861](https://doi.org/10.21105/joss.00861).
- McKay GM. 1973. Behavior and ecology of the Asiatic elephant in south-eastern ceylon. *Proceedings of Smithsonian Contributions to Zoology*; (WA), D.C. USA: Smithsonian Institution.
- Moritz P, Nishihara R, Wang S, Tumanov A, Liaw R, Liang E, Elibol M, Yang Z, Paul W, MI Jordan, Stoica I. 2018. Ray: a distributed framework for emerging AI applications. In *Proceedings of the 13th USENIX conference on Operating Systems Design and Implementation OSDI*; Carlsbad, CA, USA. 18. p. 561–577.
- Mosteller F, Tukey JW. 1968. Data analysis, including statistics. In: Lindzey G and Aronson E, editors. *Handbook of social psychology* Vol. 2, Publisher: Addison-Wesley; p. 80–203.
- Niculescu-Mizil A, Caruana R. 2005. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning (ICML)*; Bonn, Germany; 625–632. doi: [10.1145/1102351.1102430](https://doi.org/10.1145/1102351.1102430).
- Pan SJ, Yang Q. 2009. A survey on transfer learning. *IEEE Trans Knowl Data Eng*. 22 (10):1345–1359. doi: [10.1109/TKDE.2009.191](https://doi.org/10.1109/TKDE.2009.191).
- Paszke A. 2019. PyTorch: an imperative style, high-performance deep learning library. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*; Vancouver, Canada. p. 8026–8037.
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J. 2011. Scikit-learn: machine learning in python. *J Mach Learn Res*. 12:2825–2830.

- Platt J. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv Large Margin Classifiers*. 10(3):61–74.
- Poole JH. 1994. Sex differences in the behaviour of African elephants. In: Short R and Balaban E, editors. *The differences between the sexes*. 1st ed. Cambridge University Press; p. 331–346.
- Poole JH. 2011. Behavioral contexts of elephant acoustic communication. In: Moss CJ, Croze H Lee PC, editors. *The Amboseli elephants: a long-term perspective on a long-lived mammal*. University of Chicago Press; p. 125–161. [10.7208/9780226542263-011](https://doi.org/10.7208/9780226542263-011).
- Poole JH, Granli P. 2011. Signals, gestures, and behavior of African elephants”. In: Moss CJ, Croze H Lee PC, editors. *The Amboseli elephants: a long-term perspective on a long-lived mammal*. Chicago, Illinois, USA: University of Chicago Press; p. 109–124. [10.7208/chicago/9780226542263.003](https://doi.org/10.7208/chicago/9780226542263.003). 0008.
- Poole JH, Granli P. 2021. The elephant ethogram: a library of African elephant behaviour. *Pachyderm*. Vol. 62. Chicago, Illinois, U.S.: University of Chicago Press; p. 105–111. doi: [10.69649/pachyderm.v62i.462](https://doi.org/10.69649/pachyderm.v62i.462).
- Poole JH, Payne K, Langbauer WR, Moss CJ. 1988. The social contexts of some very low frequency calls of African elephants. *Behav Ecol Sociobiol*. 22(6):385–392. doi: [10.1007/BF00294975](https://doi.org/10.1007/BF00294975).
- Poole JH, Tyack PL, Stoeger-Horwath AS, Watwood S. 2005. Elephants are capable of vocal learnin. *En Nature*. 434(7032):455–456. doi: [10.1038/434455a](https://doi.org/10.1038/434455a).
- Quinlan JR. 1986. Induction of decision trees. *Mach Learn*. 1(1):81–106. doi: [10.1007/BF00116251](https://doi.org/10.1007/BF00116251).
- Rahimi A, Recht B. 2008. Weighted sums of random kitchen sinks: replacing minimization with randomization in learning. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*; Vancouver, Canada. Vol. 21.
- Ramachandran P, Parmar N, Vaswani A, Bello I, Levskaya A, Shlens J. 2019. Stand-alone self-attention in vision models by Wallach HM. editors. *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*; Vancouver, (BC), Canada; 68–80.
- Rumelhart DE, Hinton GE, Williams RJ. 1986. Learning representations by back-propagating errors. *Nature*. 323(6088):533–536. doi: [10.1038/323533a0](https://doi.org/10.1038/323533a0).
- Serizel R, Turpault N, Shah A, Salamon J. 2020. Sound event detection in synthetic domestic environments. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Online p. 86–90. doi: [10.1109/ICASSP40776.2020.9054478](https://doi.org/10.1109/ICASSP40776.2020.9054478).
- Silva MBCK. 2017. A sound processing pipeline for robust feature extraction to detect elephant rumbles [bachelor’s thesis]. Sri Lanka: University of Colombo School of Computing.
- Soltis J. 2010. Vocal communication in African elephants (*Loxodonta africana*). *Zoo Biol*. 29 (2):192–209.
- Stöger AS, Heilmann G, Zeppelzauer M, Ganswindt A, Hensman S, Charlton BD. 2012. Visualizing sound emission of elephant vocalizations: evidence for two rumble production types. *PLOS ONE*. 7(11):e48907. doi: [10.1371/journal.pone.0048907](https://doi.org/10.1371/journal.pone.0048907).
- Stone M. 1974. Cross-validatory choice and assessment of statistical predictions. *J Royal Stat Soc*. 36(2):111–133. doi: [10.1111/j.2517-6161.1974.tb00994.x](https://doi.org/10.1111/j.2517-6161.1974.tb00994.x).
- Vasconcelos C, Birodkar V, Dumoulin V. 2022. Proper reuse of image classification features improves object detection. [10.48550/arXiv.2204.00484](https://arxiv.org/abs/10.48550/arXiv.2204.00484).
- Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, AN Gomez, Kaiser Ł, Polosukhin I. 2017. Attention is all you need by Guyon I, et al. editors. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*; Long Beach, (CA), USA: Curran Associates, Inc. Vol. 30.
- Venter PJ, Hanekom JJ. 2010. Automatic detection of African elephant (*Loxodonta africana*) infrasonic vocalisations from recordings. *Biosyst Eng*. 106(3):286–294. doi: [10.1016/j.biosystem.seng.2010.04.001](https://doi.org/10.1016/j.biosystem.seng.2010.04.001).
- Williams C, Seeger M. 2000. Using the Nyström method to speed up kernel machines. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*; Denver, (CO), USA: MIT Press. Vol. 13.
- Williams C, Tiwari SK, Goswami V, de Silva S, Kumar A, Baskaran N, Yoganand K, Menon V. 2020. *Elephas maximus*. In *The IUCN Red List of Threatened Species*. doi: [10.2305/IUCN.UK.2020-3.RLTS.T7140A45818198.en](https://doi.org/10.2305/IUCN.UK.2020-3.RLTS.T7140A45818198.en).

- Wood JD, Mccowan B, Langbauer WR, Viljoen JJ, Hart LA. 2005. Classification of African elephant *Loxodonta africana* rumbles using acoustic parameters and cluster analysis. *Bioacoustics*. 15(2):143–161. doi: [10.1080/09524622.2005.9753544](https://doi.org/10.1080/09524622.2005.9753544).
- Wrege PH, Rowland ED, Keen S, Shiu Y. 2017. Acoustic monitoring for conservation in tropical forests: examples from forest elephants. *Methods Ecol Evol*. 8(10):1292–1301. doi: [10.1111/2041-210X.12730](https://doi.org/10.1111/2041-210X.12730).
- Zadrozny B, Elkan C. 2001. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of International Conference on Machine Learning (ICML)*; Williamstown, (MA), USA. Vol. 1. p. 609–616.
- Zadrozny B, Elkan C. 2002. Transforming classifier scores into accurate multiclass probability estimates. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*; Edmonton (AB), Canada: ACM. p. 694–699. doi: [10.1145/775047.775151](https://doi.org/10.1145/775047.775151)
- Zeppelzauer M, Hensman S, Stöger AS. 2015. Towards an automated acoustic detection system for free-ranging elephants. *Bioacoustics*. 24(1):13–29. doi: [10.1080/09524622.2014.906321](https://doi.org/10.1080/09524622.2014.906321).
- Zeppelzauer M, Stöger AS. 2015. Establishing the fundamentals for an elephant early warning and monitoring system. *BMC Res Notes*. 8(1):409. doi: [10.1186/s13104-015-1370-y](https://doi.org/10.1186/s13104-015-1370-y).
- Zeppelzauer M, Stöger AS, Breiteneder C. 2013. Acoustic detection of elephant presence in noisy environments. In *Proceedings of the 2nd ACM international workshop on Multimedia analysis for ecological data (ACM)*; Barcelona Spain: ACM. p. 3–8. doi: [10.1145/2509896.2509900](https://doi.org/10.1145/2509896.2509900).
- Zhao H, Jia J, Koltun V. 2020. Exploring self-attention for image recognition. *Proceedings of IEEE conference on computer vision and pattern recognition (CVPR)*; Seattle, (WA), USA: Computer Vision Foundation IEEE. p. 10073–10082. doi: [10.1109/CVPR42600.2020.01009](https://doi.org/10.1109/CVPR42600.2020.01009).
- Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H, He Q. 2020. A comprehensive survey on transfer learning. *Proceedings of the IEEE*. 109(1):43–76. doi: [10.1109/JPROC.2020.3004555](https://doi.org/10.1109/JPROC.2020.3004555).