A full end-to-end deep approach for detecting and classifying jaw movements from acoustic signals in grazing cattle

Mariano Ferrero[a], Leandro D. Vignolo[a], Sebastián R. Vanrell[a], Luciano Martinez-Rau[a], José O. Chelotti[a,b], Julio R. Galli[c], Leonardo L. Giovanini[a], H. Leonardo Rufiner[a,d]

[a] *Instituto de Investigación en Señales, Sistemas e Inteligencia Computacional, sinc(i), FICH-UNL/CONICET, 3000 Santa Fe, Argentina*
[b] *TERRA Teaching and Research Center, University of Liège, Gembloux Agro-Bio Tech (ULiège-GxABT), 5030 Gembloux, Belgium*
[c] *Instituto de Investigaciones en Ciencias Agrarias de Rosario, IICAR, Facultad de Ciencias Agrarias, UNR-CONICET, Parque J.F. Villarino, S2125 Zavalla, Argentina*
[d] *Laboratorio de Cibernética, Facultad de Ingeniería, Univ. Nacional de Entre Ríos, 3100 Oro Verde, Argentina*

Abstract

Monitoring the foraging behaviour of ruminants is a key task to improve their productivity and welfare. During the last decades, several monitoring approaches have been proposed based on different types of sensors such as pressure-based, accelerometers and microphones. Among them, microphones have been one of the most promising options because they acoustic signals provide comprehensive information about the foraging behaviour. In this work, a fully end-to-end deep architecture is proposed in order to perform both detection and classification tasks of masticatory events in one step, relying only on raw acoustic signals. The main benefit of this novel approach is the substitution of handcrafted preprocessing and feature extraction phases for a pure deep learning approach, which has shown better performance in related fields. Furthermore, different data augmentation techniques have been

Email address: `mferrero@sinc.unl.edu.ar` (Mariano Ferrero)

evaluated to address the data shortness for models development, typical in this field. The results demonstrate that the proposed architecture achieves a F1 score value of 79.82, which represents an increment close to 18% with respect to other state-of-the-art algorithms. Moreover, the proposed data augmentation techniques provide further performance enhancements, emerging as interesting alternatives in this field.

Keywords: Deep learning, data augmentation, acoustic monitoring, precision livestock farming, ruminant foraging behaviour.

---

1. Introduction

Specific changes in animal behaviour are directly related to its physical conditions (Frost et al., 1997), therefore tracking these changes comprises an essential task of livestock management monitoring. Traditionally, it has been done by manual observation, which is labour-intensive and unfeasible in some practical scenarios. With the advances in communication and information technologies, new automatic and non-invasive methods arose to boost data collection and processing, simplifying herd management tasks (Neethirajan, 2020).

Monitoring ruminants' foraging behaviour is a critical and challenging task. When long-term analyses are performed (ranging from several minutes to hours), two main activities must be distinguished: rumination and grazing. These activities are build-up on different jaw movement (JM) events: bites, chews and chew-bites (Ungar et al., 2006; Milone et al., 2012). Bites reflect the apprehension and severance of forage, and chews, the herbage comminution. A combination of them in the same jaw movement is called a chew-bite event. Monitoring the number of these events helps to provide useful information regarding animal health, nutrition status, welfare and foraging activities (De Boever et al., 1990). For example, a consistent reduction in rumination activity might indicate the presence of health disorders or diseases (Calamari et al., 2014; Paudyal et al., 2018).

Different sources of information have been used in the last decades to detect and classify JM events (Andriamandroso et al., 2016; Monteiro et al., 2021). Initially, the proposed strategy was based on observation (in-situ or video recordings), switches and jaw strap adjustment (Balch, 1958; Penning, 1983; Matsui and Okubo, 1991). This complex and fault-prone solution heavily depends on experts and is not possible to automate it, being unfeasible in large herds (Milone et al., 2009).

Other methods that recognise JM events rely on pressure sensors mounted in a halter. The RumiWatch system (Itin and Hoch GmbH, Liestal, Switzerland) is comprised of a pressure sensor and a 3D accelerometer to gather data produced during JM. This data is later analysed by a software that discriminates between chews produced during rumination, chews produced during feeding and grazing bites (Rombach et al., 2019). Although this sensor reached good performance under different conditions (Ruuska et al., 2016; Werner et al., 2018), their main limitation is the requirement of human intervention for calibration, making infeasible it use in commercial farms (Riaboff et al., 2022). Additionally, several practical issues have been reported in the use of halters (Nydegger et al., 2011) such as frequent damage when applied in loose housing systems.

On the other hand, diverse motion sensors located in different places of the animal's body have been used to determine long-term activities (rumination, grazing, resting, among others) rather than JM events (Fogarty et al., 2020; Balasso et al., 2021; Riaboff et al., 2022).

Bite events count has been addressed using pattern matching techniques from 1D accelerometer (Tani et al., 2013), 3D accelerometer (Oudshoorn et al., 2013; Giovanetti et al., 2017) and inertial measurement unit (Andriamandroso et al., 2015). Despite the fact that motion sensors provide interesting options to automatically count feeding JM (low sampling frequency and comprehensive data), the distinction between different types of events represents a challenging task from these signals and proper validation on diverse pasture and larger duration trials is still required (Ding et al., 2022). The sensitivity of this kind of sensors might introduce errors and misclassifications due to unrelated movements with JM events (ear wiggling or head turns). Furthermore, position displacements of the motion sensor affect the JM event recognition, and they are difficult to prevent in free-ranging conditions (Kamminga et al., 2018; Li et al., 2021a).

Acoustic sensors are useful for the recognition of JM events in free-ranging environments. The use of microphones allows for capturing the sounds produced by the teeth and propagated through the bones, cavities and soft tissues of the cattle's head. The analysis of these signals is a difficult task due to the presence of environmental sounds (noises) and the high computational requirements. Beyond that, they are usually preferred over pressure and movement sensors because the acoustic signals capture more information in order to perform JM events classification (Ungar et al., 2006; Martinez-Rau et al., 2022). Milone et al. (2012) developed a computational demanding method to detect and classify JM events using hidden Markov models on spectral-domain features. Navon et al. (2013) proposed a machine learning approach to separate true events (without specific classification) from background noise and silence. Chelotti et al. (2016) proposed the Chew-Bite Real-Time Algorithm, which defined a sequential system for detecting and classifying chews, bites and chew-bites using heuristic rules and temporal features. In a later work, searching for better results, the same authors proposed a system based on machine learning called Chew-Bite Intelligent Algorithm (CBIA) (Chelotti et al., 2018). Recently, Martinez-Rau et al. (2022), proposed an algorithm for robust recognition of JM events called Chew-Bite Energy Based Algorithm. It is capable of discriminating four event types: bites, chew-bites, rumination chews and grazing chews.

Automatic detection and classification systems based on sound analysis usually perform a preprocessing stage (e.g., to improve signal-to-noise-ratio) and then execute some sort of feature extraction to feed data into the classification models. The lack of an end-to-end solution introduces several potential troubles, such as dependency on specific sound recording systems and configuration, as well as difficulties to exploit potentially valuable information not encoded in manually created features. Li et al. (2021c) introduced a comparison of several deep learning (DL) architectures to classify JM events using a preprocessing phase where frequency-domain representations are extracted from raw signals. The complete workflow proposed by these authors, to generate the inputs of neural networks models includes the following steps: background noise removal using a band-stop filter, uninformative data removal based on manually created thresholds and Mel-frequency cepstral coefficients calculation. Compared with traditional machine learning techniques, the use of DL models brings the opportunity to automatically discover patterns and features from data at the expense of higher computational costs.

Based on the analysis of previous research it is possible to state that DL models have been used only to classify JM events. Therefore, the application of DL models to perform JM events recognition (which involves JM events detection and the posterior classification of them), has not been explored yet. Additionally, the rest of the traditional alternatives (such as the CBIA system) heavily depend on manual feature extraction methods and arbitrarily defined pre-processing steps. Promising results presented by Li et al. (2021c) highly motivate the study of DL architectures to tackle the limitation of JM events recognition.

In this paper, a truly end-to-end approach is proposed to process raw audio signals toward the detection and the classification of JM events (bite, chew and chew-bite). The proposed DL strategy combines the power of convolutional networks for feature learning with the time modeling capabilities of recurrent units, to implement **detection and classification tasks in one step**. Several architectures have been explored and compared to point out the benefits and limitations of the proposed approach. Additionally, different data augmentation techniques have been evaluated to improve the generalisation capabilities of the proposed approach. Experimental results show the benefits of the application of the proposed deep architectures over traditional machine learning approaches. The main contributions of this paper are the following: a) a novel deep-learning model that combines convolutional and recurrent neural networks is presented. It automatically learns the features representations and the temporal dependencies between JM events from raw audio signals. b) The proposed model is able of solving the JM events detection and classification tasks in one step from raw from acoustic signals; and finally c) different data augmentation techniques were analysed to undertake the data-shortness problem.

2. Material and methods

In this article, a novel deep-learning architecture called **Deep sound** is proposed. It is based on the combination of two types of neural networks: Convolutional Neural Networks (CNN) (Lecun et al., 1998) and Recurrent Neural Networks (RNN) (Rumelhart et al., 1986). In the following sections, a brief introduction to these architectures is provided. Then, a detailed description of the proposed method is presented.

2.1. CNN and RNN

Convolutional Neural Networks (CNN) (Lecun et al., 1998) are one of the most widely used architectures for classification problems where input data comes from unstructured sources - images (Kokalis et al., 2020) or audio (Ramirez et al., 2022), for example. They are usually composed by several convolutions layers, each one containing one or more filters. In the learning stage, filters' weights (used in traditional convolution mathematical operations) are adapted in order to approximate outputs using optimisation strategies like stochastic gradient descent or back-propagation (Rumelhart et al., 1986). By doing this, the layers are capable of learning different high and low-level patterns without domain knowledge supplied.

In CNN, convolutional layers are used in combination with pooling, batch normalisation and dense layers. Pooling layers apply simple mathematical operations (such as maximum extraction) in order to reduce dimensionality, and they are commonly used after convolutional layers. On the other hand, batch normalisation layers scale the inputs, to the desired values, to accelerate the training process. Finally, dense layers correspond to a flat set of hidden neurons fully connected (FNN) with the outputs of previous layers, providing to the CNN with the ability to adapt the effect of intermediate representations, learned by convolutions, on the output. The relation between convolution with other layers is created using a flattening operation, which transforms the output of convolution layers into a vector. An important operation used in these layers (except for batch normalisation) is called **drop-out**, which introduces random crops between layer connections during the training phase to avoid model over-fitting (Hinton et al., 2012).

Recurrent Neural Networks (RNN) (Rumelhart et al., 1986) are broadly used in a wide variety of problems involving temporal sequences (Lim et al., 2019; Li et al., 2021b). RNN connects layer outputs as inputs to the same layer, enabling temporal data flow more efficiently across the network. More sophisticated architectures have been developed in recent years to overcome some RNN limitations. Gated Recurrent Units (GRU) are composed of several neurons called **cells**, each one uses two different gates: reset and update (Cho et al., 2014). These gates, tuned during the training process, allow every neuron to control the trade-off between how much information is used from previous and current states. GRU networks are composed of several GRU cells placed sequentially. A variation of a RNN proposed by Schuster and Paliwal (1997) is called Bidirectional RNN. This network introduces two identically RNN in terms of architecture, one trained with time sequences forwards and the other one with the same sequences backward, both connected to the next layer of the network. Specifically, bidirectional GRU (BGRU) achieved very promising results in sound events detection (Lu et al., 2018; Meng et al., 2022) and classification (Zhu et al., 2020).

2.2. Deep sound

Different variations of several deep architectures were studied for this problem, based on previous research in related fields (Khamees et al., 2021; Bahmei et al., 2022; Petmezas et al., 2022). The alternatives were evaluated from a theoretical perspective and the most promising ones were implemented. Thus, a hybrid one-dimensional (1D) CNN-BGRU network architecture is proposed, named **Deep sound**. To the best of the authors' knowledge, this represents the first deep end-to-end approximation to the problem of JM events detection and recognition from acoustic signals. The network receives the sound windows extracted from the original audio files without any prior preprocessing or feature extraction phase, and classifies them into one of four possible classes: chew, bite, chew-bite and no-event. Therefore, the proposed method tackles the problems of JM event detection and classification at the same time.

The proposed model structure is given by: an input layer and several hidden layers distributed in three main blocks corresponding to CNN, BGRU, and FNN. An overall schematic of the proposed model is presented in Figure 1(a), while a detailed description of the architecture is showed in Figure 1(b). The first part of Figure 1(b) represents the CNN block of the model,

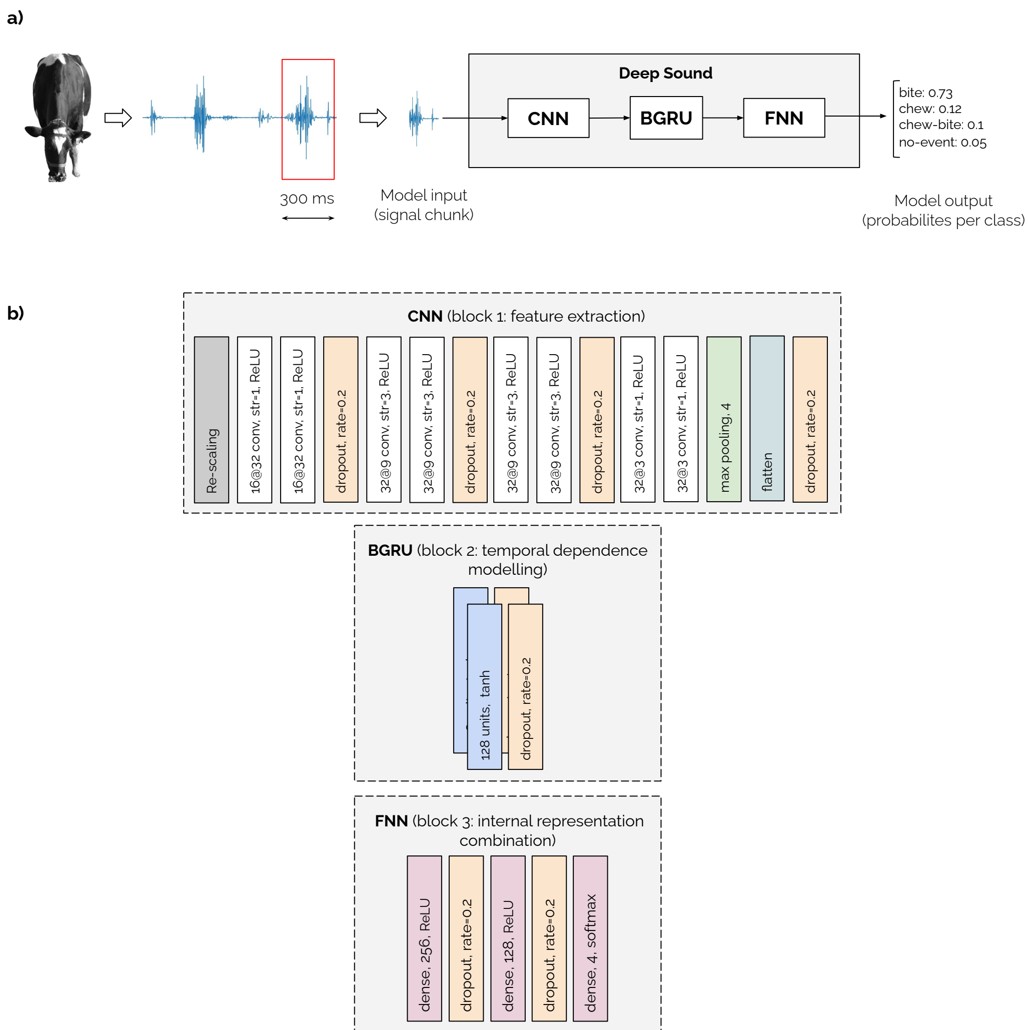

Figure 1: The overall proposed method architecture. a) Input signals correspond to audio chunks extracted using fixed-length time windows and passed through the CNN (first block) to automatically extract features. The output of this block is passed to the bidirectional GRU to capture temporal dependencies in data. Finally, the output of the second block is fed into the FNN block, combining information in dense layers, and predicts class probabilities for each input sample. b) Specification of layers in each block, including the number of filters or units, filter size (for convolutional layers), and activation functions.

which is a combination of 1D convolutional layers, dropout operations, and max pooling layers. This way, the network is capable of extracting low- and high-level features from audio chunks and performing dimensionality reduction at the same time. At the beginning of this block, a re-scaling layer adapts the range of input values for implementation purposes. A flatten operation is also used to create a raw vector from the last convolutional layer. A complete definition of layer configurations, such as number of filters and filter sizes, is provided in the figure. The second block in Figure 1(b) introduces a recurrent network, composed of a BGRU layer with 128 cells. The purpose of this block is to capture time dependencies in the data. The last block of the network implements a typical FNN with three dense layers and two dropout operations. Blocks one and three are placed into time-distributed wrappers, allowing the same layers to be applied to each window of the input signals. This means that the same set of connection weights is trained and used in these blocks for every time window. All convolutional layers use the activation function rectified linear unit (ReLU), whilst the cells of the BGRU use hyperbolic tangent and sigmoid. The first and second dense layers perform both ReLU, and the last dense layer uses the soft-max function for classification. All layers (convolutional, recurrent and dense) use the Xavier initialisation method (Glorot and Bengio, 2010) and bias terms were initialised to zero.

The main limitations of the proposed method are: a) a considerable amount of labelled data is needed for training, b) the interpretability of the method and its outputs is limited (Arrieta et al., 2020; Hoxhallari, 2022), and c) a considerable amount of processing is required in the inference phase.

2.3. Acoustic dataset

2.3.1. Original dataset

The data used in this work is one of the first open datasets in this field of study (Vanrell et al., 2020). The fieldwork to obtain this dataset took place at the Campo Experimental J.F. Villarino, Facultad de Ciencias Agrarias, Universidad Nacional de Rosario, Zavalla, Argentina. The recordings include sounds produced by dairy cows in individual grazing sessions conducted over a 5-day period. Microphones used to record audio signals (Nady 151 VR, Nady Systems, Oakland, CA, USA) were located on the cow's forehead and covered with rubber foam. Further details about experimental design could be found in the dataset article (Vanrell et al., 2020).

A total of 52 raw audio signals (WAV audio files, mono, 16-bits, 22.05 kHz) are available [1]. A summary of the dataset contents is presented in Table 1. Each audio signal consists of sequences of JM events – bites, chews, and chew-bites – separated by silence (ranging from 19 to 152 s, average duration 62.76 ± 28.61 s). Two different experts in ruminant foraging behaviour independently performed the identification of each JM (including event label, start, and end time) by analysing videotapes and sounds at the same time. Agreement results were 100% for bites, 98.2% for chews, and 99.1% for chew-bites. There were 2.7% of insertions and 0.9% of deletions. Thus, the total segmentation and classification accuracy was 93.6%. Both experts worked together to achieve a final decision in case of disagreement.

[1] Direct URL to data: https://github.com/sinc-lab/dataset-jaw-movements

Table 1: Summary of audio files grouped by pasture and height.

Pasture	Height	Chews	Bites	Chew-Bites	Overall duration
Alfalfa	Tall	416	148	322	14 min 26 s
Alfalfa	Short	260	179	123	12 min 42 s
Fescue	Tall	487	100	238	14 min 03 s
Fescue	Short	454	94	217	13 min 13 s
Total		1617 (53%)	521 (17%)	900 (30%)	54 min 24 s

2.3.2. Data preparation

Since the delimitation of most of the labels in the original dataset was inaccurate with respect to the actual JM events, an improvement to label bounds has been proposed in the present work. Conducting a visual inspection of original signals and labels, it is possible to notice that there is not a perfect time delimitation between JM events presence and timestamps. Figure 2 shows some examples where over estimations of JM events duration are introduced. To tackle this situation, time event delimiters have been adapted using a label erosion method based on signal envelope computation and selected thresholds. The events start timestamp was moved to the position where the signal envelope reaches a certain threshold; similarly, this process was repeated in the opposite direction with the event end timestamp, generating a time shift respecting the original label.

The threshold is defined as follows: after JM event envelope calculation, the maximum value is obtained and multiplied by a factor adapted to the differences between event characteristics. Table 2 introduces start and end factors applied to different event classes.

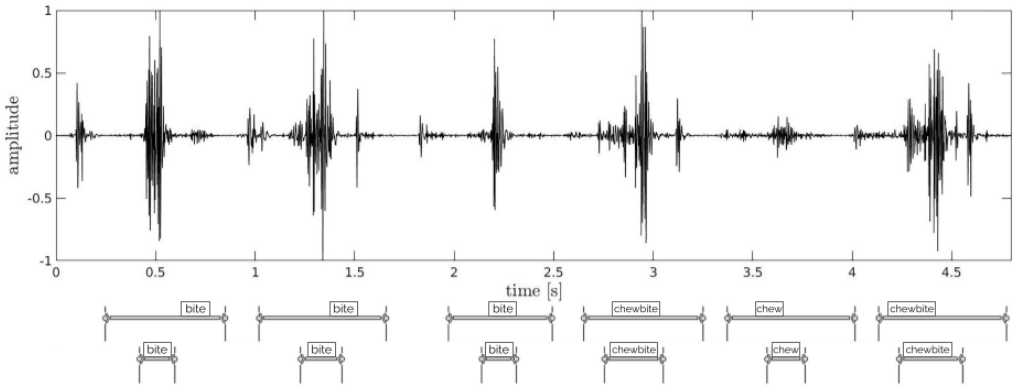

Figure 2: Visual comparison of an example of a signal with original (top) and eroded (bottom) labels with time delimiters (timescale on the top is expressed in seconds).

Table 2: Scale factors applied to maximum values extracted from the signal envelope to define threshold calculation.

JM event type	Start factor	End factor
Bite	0.4	0.4
Chew	0.5	0.5
Chew-Bite	0.15	0.4

Original audio signals have been recorded at 22.05 kHz. In order to reduce dimensionality and computational costs, all files were downsampled to 6 kHz. In addition to this, original audio signals were divided into small chunks of data using sequentially ordered windows. Different window sizes have been evaluated during the initial experimentation, considering the average duration of JM events, and the value of 300ms produced the best results, with a hop length of 150 ms. The average duration of the JM events is 330 ms ($\pm$ 150 ms), which means that two consequent windows might be needed to represent one JM event. To assign a label to a particular signal window, a minimum overlapping of 40% with a JM event label is required, guaranteeing that if only a small part of a window corresponds to a JM event of interest (bite, chew or chew-bite) it is tagged as 'no-event'.

2.3.3. Data augmentation

A distinctive characteristic of the proposed approach is the number of parameters to be learned or tuned during the training process. Consequently, the use of a small amount of data may lead to overfitting. In the context of precision livestock farming, and JM events recognition in particular, getting more annotated signals requires great effort and resources. To overcome this problem, data augmentation techniques are traditionally employed to artificially create synthetic samples from original ones (Nanni et al., 2021; Bahmei et al., 2022). Despite that data augmentation is well-known for image-related problems (Shorten and Khoshgoftaar, 2019), custom techniques are usually required when working with audio signals.

When new samples are created from existing data, two facts should be considered: *i)* the types of perturbations applied on original data to create a different one, but still usable synthetic audio signal (named here **augmentation technique**), and *ii)* how to apply them to every training sample (**augmentation protocol**). Several augmentation techniques have been explored in early experimentation (including but not limited to loop, pitch shift, time stretch and percussive). Finally, six data augmentation techniques were selected:

- Resynthesis by Linear Predictor Coefficients (LPC): given an input signal, the LPC is estimated, randomly perturbed, and finally used to generate a new signal using a resynthesis process.

- Reverse: a copy is created from original values by doing a backward pass.

- Random crop: randomly pick a very small fraction (1%) of continuous values from the input signal and turn them to zero.

- Background noise: add white noise to the original signal, using a signal-to-noise ratio of 10 dB.

- Amplitude change: increase or decrease signal amplitude by a certain decibel amount. Positive values stand for increases, while negative stands for amplitude decrease.

- Frequency filters: apply a second-order Butterworth high-pass or low-pass filter to the input signal. The high-pass and low-pass filters have a cut-off frequency of 500 Hz and 100 Hz, respectively.

On the other hand, two different augmentation protocols were tested:

- Random: pick one augmentation technique and use it to generate a synthetic signal.

- Serial: create a pipeline serialising all defined augmentation techniques in order to apply them one by one. This way the input to the first technique is the original audio signal and its output is fed to the subsequent technique.

During experimentation, three synthetic signals were created from every single input sample when defining an augmentation protocol. These values were selected in order to explore the effect of this component without significantly affecting the computational cost.

2.4. Experimentation methodology

2.4.1. Model selection approach

For all experiments, the models were evaluated using 10-fold cross-validation (CV). Every fold contains 5 or 6 input files, randomly selected from the total of 52 available. In this way, every input file was included in only one fold. In addition to this, 20% of the 9 folds used for training on every iteration were reserved for validation. The assignment of sound files to the train and test sets in each fold was fixed across different experiments. The number of windows in test sets was 2168 ± 360 (proportion per class: $5 \pm 1\%$ bites - $18 \pm 1\%$ chews - $14 \pm 4\%$ chew-bites - $63 \pm 4\%$ no-event). The number of windows in train and validation sets changed from one experiment to another due to the use of different data augmentation configurations. The training samples were weighted in order to tackle classes imbalance according to the following expression:

$$cw_{ic} = n_{max}/n_c, \tag{1}$$

where cw_{ic} is the class weight of instance i of class c, n_{max} is the number of instances of the majority class and n_c is the number of instances of class c. Finally, the experiments were set-up with a total of 1500 epochs with early stopping (50 epochs tolerance), Adam (Kingma and Ba, 2014) as the optimizer, the batch size was fixed to 10, 0.001 as the learning rate, and categorical cross entropy as loss function. Default values were used for the remaining parameters.

2.4.2. Evaluation metrics

The dynamical problem of simultaneous detection and classification of JM events using raw audio signals is substantially different from the approach of dividing the problem into JM event detection and subsequent classification based on previously detected events (Chelotti et al., 2018; Martinez-Rau et al., 2022). In the former, the temporal component plays a very important role, since the need to properly detect JM event's onsets and offsets affects the results of the classification. Based on this, the generation of a model that deals with detecting and classifying events at once requires the use of a validation mechanism that is capable of considering aspects related to temporality, as well as predicted labels accuracy.

To evaluate JM events detection and classification performances, the sed_eval standardised toolbox was used (Mesaros et al., 2021). It is a transparent and broad library to evaluate sound event recogniser systems. The toolbox was designed for the task of sound event recognition, which involves locating and classifying sounds in audio recordings, estimating onset and offset for distinct sound event instances and providing a textual descriptor for each. This matches the task presented in this work, where sound JM events classes are chew, bite and chew-bite. A temporal tolerance (collar) of 300 ms was used. This value was determined based on preliminary experimentation considering two main aspects: 1) the collar should be smaller than the average event duration (330 ms) in order to ensure overlap between reference and predicted window. 2) it should avoid undesired overlap between two adjacent events (with an average separation between two adjacent events of 726 ms). The selected value meets both criteria.

With the use of sed_eval toolbox, a reference JM event is correctly detected if two conditions are met: *i)* The start timestamp of the predicted JM event is located in the interval defined by reference onset ± tolerance value. *ii)* The end timestamp of the predicted JM event is located in the interval defined by reference offset ± tolerance value. Figure 3 introduces a graphical representation of how this toolbox works.

Based on before mentioned evaluation toolbox, several well-known metrics have been used:

$$precision = \frac{TP}{TP + FP},$$

$$recall = \frac{TP}{TP + FN},$$

$$F1\ score = \frac{2 * precision * recall}{precision + recall},$$

$$error\ rate = \frac{S + D + I}{N}$$

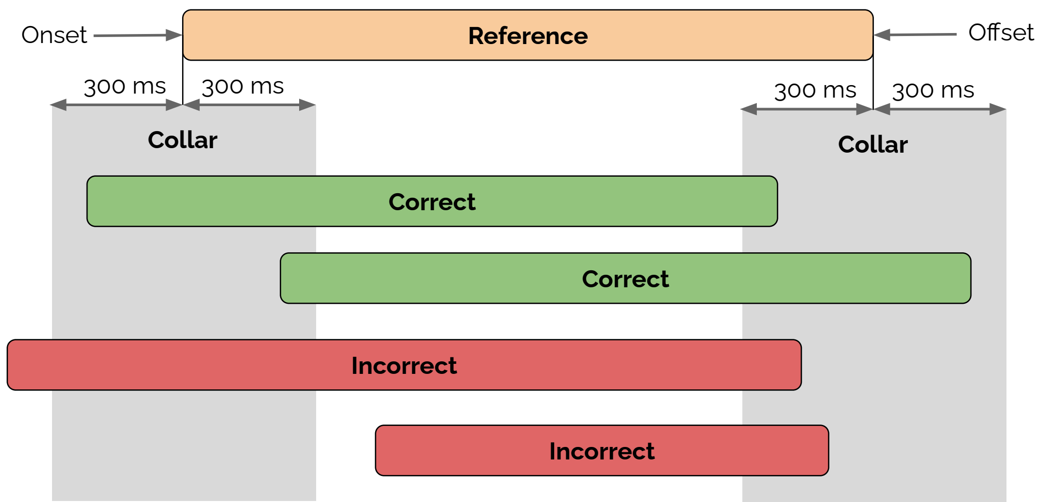

Figure 3: Illustration based on Mesaros et al. (2021) where two correct and two incorrect predicted JM events are presented, compared with a reference JM event using a tolerance value of 300 ms.

where TP denotes true positive, FP false positive, FN false negative, S substitutions (correct detected JM events in system output but incorrectly labelled), I insertions (detected events from system output which do not exist in the ground truth), D deletions (ground truth events which are not detected) and N is the total number of reference events. Due to the presence of class imbalance in the original dataset, JM events distributions are taken into account to calculate average final results. When using this approach for metrics calculation micro averages were computed (Sokolova and Lapalme, 2009), which means that TP, FP and FN are calculated by summing up samples through all classes. For example, the term TP is finally expressed by $TP_c + TP_{cb} + TP_b$, representing the amount of TP for chews, chew-bites and bites, respectively.

2.5. Experimental setup

The design and implementation of the proposed model were developed using Python 3.6.2 and TensorFlow-GPU 2.6.2. Different utilities from the Python library scikit-learn 0.24.2 have been used, such as label encoders and k-fold extraction. Augly (Papakipos and Bitton, 2022), a Python data augmentation library, was used to apply some of the previously mentioned augmentation techniques (background noise, amplitude change and frequency filters). Experiments were performed using an Intel® Core™ i7-8700 3.20GHz CPU, 64 GB RAM and 24 GB NVIDIA GeForce RTX 3090 GPU. A Titan XP GPU was also used for model exploration, preliminary experimentation and hyperparameter tuning.

3. Results

During the optimisation process, a total of 39 experiments were tested, aiming to find the best model architecture configuration considering variations in the CNN part of the model (block 1 in Figure 1). The most promising and standard hyper-parameters combinations (such as the number of layers, number of filters, and dimension of filters) have been considered for this exploration. All experiments used the 10-fold CV method described in Section 2.4.1. Layers configuration from most representative experiments are described in Figure 4, and their respective recognition results are presented in Table 3. In terms of performance, architecture (c) exhibited the highest F1 score value. Moreover, this model also reached the lowest error rate. Therefore, it is possible to establish that architecture (c) configures the best combination explored, considering numbers of layers, number of filters and

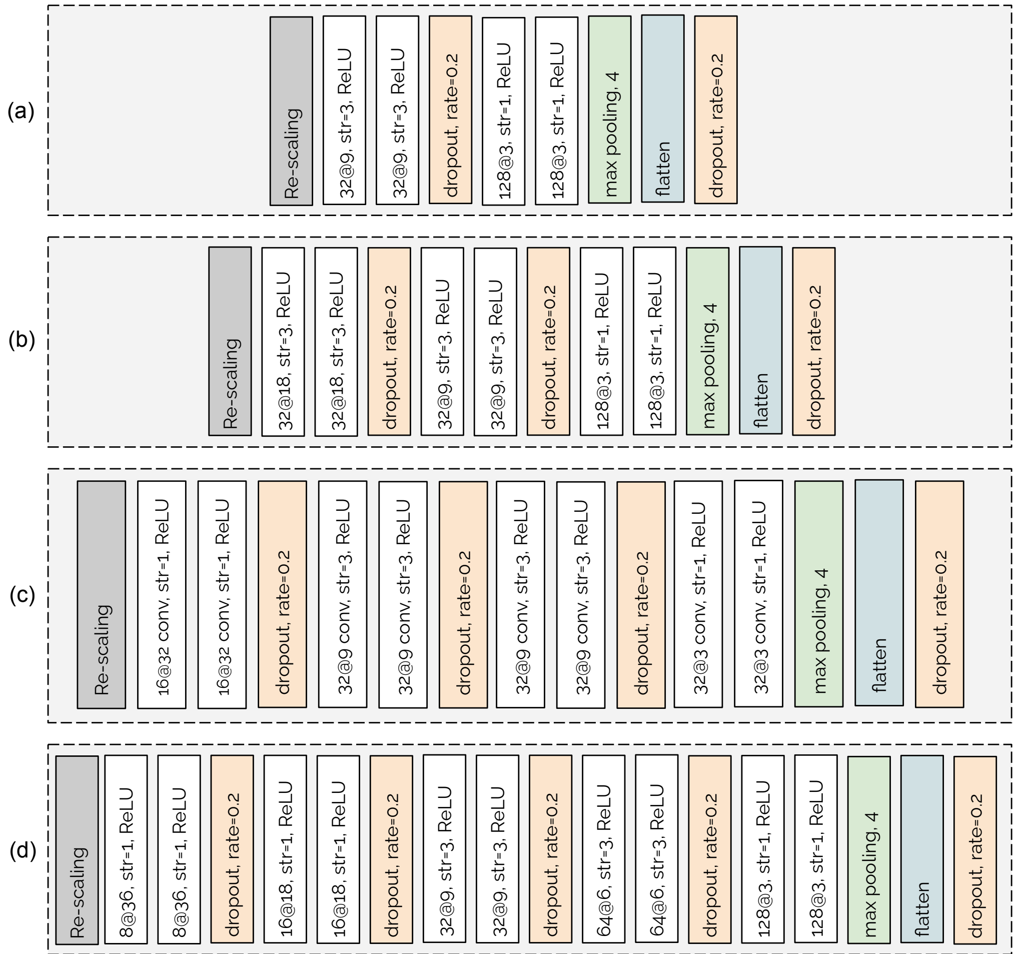

Figure 4: Different CNN architectures used for exploration. Convolution layers definition consist of number of filters, filter size and stride. No padding method was used.

filter dimensions.

As described previously, the proposed model is composed of three blocks with different types of layers. Table 4 exhibits the performance of the proposed model without using the RNN (block 2 in Figure 1). It can be seen that providing the capacity to capture temporal relationships in acoustic sequences gives a significant advantage to the network.

In addition to the optimisation of model hyperparameters, an exploration

Table 3: Recognition results of the proposed model for different layers architectures on the CNN block. For every experiment, average values and standard deviation of 10-folds CV are presented.

	Precision ↑	Recall ↑	F1 score ↑	Error rate ↓	Deletion ↓	Insertion ↓
(a)	63.13 ± 6.53	79.81 ± 6.06	70.45 ± 6.26	0.54 ± 0.12	0.07 ± 0.03	0.34 ± 0.07
(b)	71.91 ± 5.26	85.77 ± 3.37	78.19 ± 4.33	0.39 ± 0.08	0.05 ± 0.02	0.25 ± 0.06
(c)	**73.72 ± 4.92**	**87.16 ± 2.74**	**79.82 ± 3.70**	**0.37 ± 0.08**	**0.05 ± 0.01**	**0.24 ± 0.07**
(d)	73.38 ± 5.30	85.92 ± 3.81	79.12 ± 4.46	0.37 ± 0.09	0.06 ± 0.02	0.23 ± 0.06

Table 4: Evaluation of the impact of the RNN block in the proposed model. For each experiment, the average and the standard deviation of 10-fold CV are presented.

	Precision ↑	Recall ↑	F1 score ↑	Error rate ↓	Deletion ↓	Insertion ↓
Deep sound	**73.72 ± 4.92**	**87.16 ± 2.74**	**79.82 ± 3.70**	**0.37 ± 0.08**	**0.05 ± 0.01**	**0.24 ± 0.07**
Deep sound (no RNN)	48.77 ± 3.89	82.55 ± 3.64	61.26 ± 3.79	0.95 ± 0.14	0.07 ± 0.03	0.77 ± 0.12

of the impact of using several data augmentation techniques and protocols were carried out using the proposed Deep sound (c) architecture. Table 5 introduces the results of different experiments using isolated augmentation techniques (in order to measure the individual impact) and combining many of them at the same time with a particular augmentation protocol. The protocol combined the three best individual techniques based on its F1 score (background noise, random crop and amplitude (+2 dB)) to form a top 3 augmentation technique. This combination has been tested using serial and random protocols. The highest F1 score (p=0.006; Wilcoxon signed-rank test) (Wilcoxon, 1945) was reported using the top 3 augmentation techniques with serial augmentation protocol.

Finally, a contrast between the proposed model and other state-of-the-art

Table 5: Results of the proposed model using different augmentation techniques and protocols. For each experiment, the average and the standard deviation of 10-fold CV are presented. The number of copies generated per original sample was fixed to three.

Augmentation technique	Augmentation protocol	Precision ↑	Recall ↑	F1 score ↑	Error rate ↓
No augmentation	-	73.72 ± 4.92	**87.16 ± 2.74**	79.82 ± 3.69	0.37 ± 0.08
LPC	-	71.88 ± 4.72	86.67 ± 2.64	78.54 ± 3.69	0.40 ± 0.08
Background noise	-	76.83 ± 5.61	85.71 ± 3.46	80.96 ± 4.37	0.32 ± 0.09
Random crop	-	77.28 ± 7.72	86.31 ± 3.72	81.43 ± 5.63	0.32 ± 0.12
Amplitude (+2 dB)	-	76.14 ± 5.33	86.60 ± 3.89	80.98 ± 4.37	0.33 ± 0.08
Amplitude (-2 dB)	-	74.24 ± 6.45	86.18 ± 3.33	79.68 ± 4.78	0.37 ± 0.10
High-pass filter	-	70.63 ± 5.57	85.25 ± 3.82	77.19 ± 4.59	0.42 ± 0.09
Low-pass filter	-	66.64 ± 8.37	83.80 ± 4.99	74.09 ± 6.83	0.50 ± 0.17
Reverse	-	72.90 ± 5.91	86.78 ± 2.61	79.16 ± 4.38	0.39 ± 0.09
Top 3	Serial	**78.39 ± 4.09**	86.60 ± 3.08	**82.27 ± 3.42**	**0.29 ± 0.06**
Top 3	Random	77.04 ± 5.45	87.06 ± 3.19	81.67 ± 3.99	0.32 ± 0.08

methods has been carried out. In particular, the algorithm called Chew-Bite Intelligent Algorithm (CBIA) (Chelotti et al., 2018) and an implementation of the ResNet proposed by Hershey et al. (2017) for raw audio classification were compared using the same evaluation toolbox and metrics. The CBIA method was selected because it offers the best results of state-of-the-art in the detection and classification of JM events problem (unlike Li et al. (2021c), where only classification is performed) for chew, bite and chew-bite labels. Moreover, as the authors mention in their work, the Li et al. (2021c) proposal does not offer improvements in terms of classification rates with respect to Chelotti et al. (2018) approach. The ResNet architecture was selected because it is one of the best well-known DL models proposed for image classification and reached the best results for audio classification tasks (Hershey et al., 2017) among other DL models (such as VGG (Simonyan and

Table 6: Comparison between the proposed method and other state-of-the-art algorithms, CBIA and ResNet architecture.

	Precision ↑	Recall ↑	F1 score ↑	Error rate ↓	Deletion ↓	Insertion ↓
Deep sound	**78.39 ± 4.09**	**86.60 ± 3.08**	**82.27 ± 3.42**	**0.29 ± 0.06**	**0.06 ± 0.02**	0.17 ± 0.05
CBIA	68.69 ± 7.56	70.30 ± 7.92	69.43 ± 7.52	0.42 ± 0.11	0.10 ± 0.05	**0.12 ± 0.06**
ResNet audio	43.99 ± 12.96	54.99 ± 23.35	47.9 ± 17.16	0.97 ± 0.27	0.3 ± 0.21	0.52 ± 0.2

Zisserman, 2014), Inception (Szegedy et al., 2016) or AlexNet (Krizhevsky et al., 2017)).

The results of this comparison are presented in Table 6 and separated by class in Table 7. Deep sound refers to the best architecture configuration (architecture (c)), trained using top 3 (background noise, random crop and amplitude increase +2 dB) serial augmentation protocol. It can be seen that there is a significant improvement using the proposed algorithm (p=0.002 based on F1 score; Wilcoxon signed-rank test) (Wilcoxon, 1945). Despite this, results from all methods are higher for chew events, probably related to the fact that this is the most predominant class. Regarding deletion metric, the proposed algorithm increases the number of ground truth events detected. However, CBIA presents a smaller number of insertions than the proposed algorithm.

Finally, a summary of the different approaches is introduced in Figure 5. In terms of F1 score and precision, the proposed architecture (Deep sound) using augmentation techniques obtained the best results, whereas ResNet architecture led to the lowest value. On the other hand, based on the recall metric, the proposed architecture without augmentation techniques presented the best results and ResNet produced the worst. It is possible to note

Table 7: Class based results obtained for the proposed architecture and other state-of-the-art algorithms, CBIA and ResNet architecture.

	Class	Precision ↑	Recall ↑	F1 score ↑
Deep sound	Bite	73.59 ± 8.49	76.10 ± 9.16	74.27 ± 6.52
	Chew	82.56 ± 6.32	90.61 ± 3.58	86.33 ± 4.78
	Chew-Bite	73.81 ± 8.40	86.53 ± 4.38	79.31 ± 5.24
CBIA	Bite	48.77 ± 10.72	66.41 ± 10.37	55.06 ± 7.48
	Chew	77.30 ± 6.59	76.69 ± 5.72	76.77 ± 4.60
	Chew-Bite	70.77 ± 15.06	60.78 ± 18.09	63.74 ± 16.65
ResNet audio	Bite	36.72 ± 20.8	55.18 ± 23.95	42.6 ± 20.7
	Chew	51.31 ± 26.02	52.6 ± 34.18	48.91 ± 28.54
	Chew-Bite	41.62 ± 12.97	62.94 ± 20.09	46.87 ± 11.95

that ResNet also exhibited higher deviations in all presented metrics.

4. Discussion

4.1. End-to-end model architecture

Based on the presented results, the use of a deep end-to-end approach provides the model the capacity to learn relevant internal representations starting from raw signals. Manual feature computation and extraction are difficult tasks, which involve a deep understanding of the studied phenomena as well as the capacity to apply that knowledge properly. This limitation is overcome in the proposed model, resulting in a significant improvement compared with traditional machine learning algorithms. It is important to highlight that the use of recurrent layers introduces a substantial benefit to the model architecture. The use of different gates allows these layers to learn

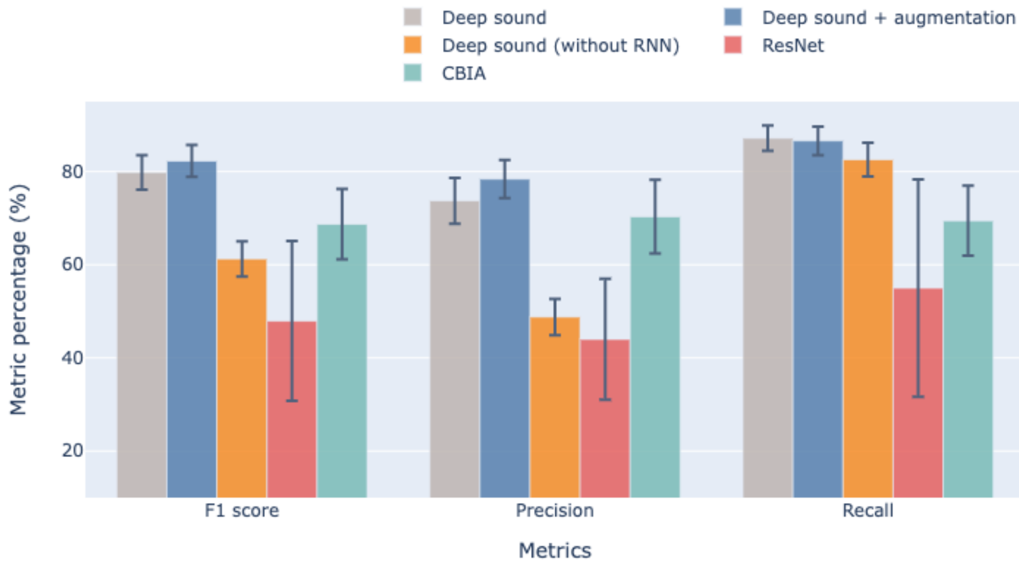

Figure 5: Overall comparison of the results obtained by the most relevant experiments of each of the presented approaches.

how much information to incorporate into their internal memory regarding new events and how much to remember from previous events. A positive impact seems reasonable based on this, attending to ruminant foraging behaviour activities, in which sometimes a single bite is followed by a sequence of chew and chew-bite events during grazing.

Regarding the model architecture, the results suggest that the use of several layers is advantageous. When using a reduced number of convolutional layers (less than 6), the recognition performance of the network is remarkably damaged. In contrast, when using at least 6 convolutional layers the model performance seems to approach similar levels. A possible explanation of this fact is that the model requires a minimum number of layers in order to extract a relevant representation from data. In terms of the number of parameters, the model architecture presented in Figure 4 (c) uses 320,229. This value probably represents a considerable increment compared to other traditional methods. However, the use of convolutional layers prevents a bigger increase in this number with respect to other neural network architectures, which use mainly dense layers.

The evaluation with different data folds shows a considerable level of deviation in the performance metrics. This effect might be due to the fact that several signals are particularly different from the rest in terms of duration (shorter) and JM events distribution (most of the present events correspond to the same class along the signal). The recognition performance decreased on those signals in all performed experiments.

4.2. Effect of learning from synthetic data

In order to increase the size of the dataset available for training in each fold, eight different data augmentation techniques were proposed and analysed (Table 5). Results showed that a subset of them allowed the model to improve the recognition performance in terms of F1 score. When analysing precision and recall separately, it is possible to note that introducing synthetic data to the training process reduces the number of detected events in general. Despite this, for some techniques there was an improvement in the precision of predictions. The results highlight the importance of using augmentation techniques to increase the generalisation capacity of the model.

Some individual techniques showed a positive impact on the performance, while others showed no impact or even a negative impact. The techniques of both low- and high-pass filters and reverse degraded the performance compared to the no augmentation approach. In contrast, when adding background noise or random crops, the model presented improvements regarding recognition results.

A comparison between proposed protocols and individual techniques highlighted that generating new samples by applying a selection of the best individual techniques, in a sequential one-by-one pipeline, is more convenient than randomly picking one of them.

4.3. Comparison against existing methods

Results presented in Table 6 and Table 7 exhibit a considerable improvement of the proposed method against the CBIA and ResNet methods in terms of recognition performance. The results obtained by the ResNet are poor in this context. This may be mainly due to the fact that the model was originally intended to process images, and it lacks capabilities to learn from temporal sequences as needed for this particular problem. It is important to note here that results reported by Chelotti et al. (2018) are affected by the use of a different tool to compare ground truth values against model predictions. In this case, the temporal alignment of both events (real and predicted) is considered using a gap or collar. By doing this, for example, a sequence of events predicted in the correct order is not considered successful if the temporal localisation does not match. Consequently, it is possible to state that the comparison method proposed in this study is more rigorous and appropriate for problems of JM event detection and classification.

In terms of computational costs, the proposed method involves a total of 464,919,007 floating point operations (FLOPs) in order to analyse one second of the signal. The details about estimation of these costs are presented in the Appendix A. This number represents an increase in the calculations needed against the CBIA (1.000:1), which needs 398,860 FLOPs to process one second of the signal. This value was estimated using the calculations reported by the authors for the version (Least Mean Squares filter and Multi-Layer Perceptron) and sampling frequency (22.05 kHz) used in the implementation conducted here. Although the proposed method represents an increase in the number of operations, the improvements obtained with respect to more accurate recognition results represent a considerable advantage in the context of applications where real-time operation is not required. The key advantage of the proposed method is its ability to accurately classify JM using raw audio signals, without any previous definition of sound features to be analysed by the system. In this stage, the computational cost of algorithms is not relevant compared with their ability to extract the appropriate information without an “expensive”, handcrafted and generally non-optimal feature engineering stage. This fact implies that this type of model can be used in the development stages of a system when relevant features for JM recognition of the sound are explored.

The interpretability of a proposed solution is another subject that must be analysed from a practical point of view. In this sense, the method presented in this paper poses a disadvantage when compared to traditional methods that use "white box" models.

On the other hand, when algorithms must be deployed on IoT systems, computational cost is a central issue since they must minimise the use of energy. This type of operational condition requires that algorithms must be optimised from the processor’s perspective, minimising the amount of energy and memory as well as the notation used to represent the information. In this way, handcrafted feature algorithms might require less implementation effort in these scenarios. The price paid is the time and work required to develop the system.

Concerning other DL methods, Li et al. (2021c) reported 88.8, 88.9 and 88.8 for F1 score, precision and recall respectively. Even though these values seems to overcome the proposed Deep sound architecture in the classification task, detection is disregarded in that study. Moreover, the limitations of the approach proposed by Li et al. (2021c), plus the evaluation metrics proposed here, should be considered in order to perform a direct comparison between both methods. Finally, it is important to note that results reported by Li et al. (2021c) slightly outperformed or was comparable to CBIA.

5. Conclusions

In this study, a novel end-to-end architecture for detection and classification of ruminant masticatory JM events was presented and evaluated with real data. The model combines two well known neural network types into a single model, generating a CNN-RNN final architecture. Different numbers of convolutional layers in the CNN block of the network have been explored. The highest recognition performance (micro F1 score up to 79.8%) was obtained using 4 pairs of convolution (plus dropout) layers. The use of data augmentation has been evaluated, which resulted in an improvement of recognition performance (almost 2.5% in terms of micro F1 score) when using a selected subset of techniques to generate synthetic samples. The proposed architecture outperformed a previous method (CBIA) by at least 10% (micro F1 score) and a ResNet implementation by more than 30% (micro F1 score). On the other hand, the proposed architecture automatically extracts features from raw signals, which introduces very promising results when compared to traditional methods that use manually created characteristics.

Future research will focus on the optimization of computational cost of the proposed method, and the analysis of its impact on recognition results. The interpretation of learned features and their corresponding qualitative analysis will be part of future works. Finally, an exploration of transfer learning, semi-supervised learning and related approaches will be studied in order to evaluate other alternatives for small quantities of labelled data.

Acknowledgments

This work has been funded by Universidad Nacional del Litoral, CAID 50620190100080LI and 50620190100151LI, Universidad Nacional de Rosario, projects 2013-AGR216, 2016-AGR266 and 80020180300053UR, Agencia Santafesina de Ciencia, Tecnología e Innovación (ASACTEI), project IO-2018–00082, Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), project 2017-PUE sinc(i). Authors would like to thank the dedication and perceptive help by Campo Experimental J. Villarino Dairy Farm staff for their assistance and support during the completion of this study. Authors also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan XP GPU used for this research.

References

Andriamandroso, A., Bindelle, J., Mercatoris, B., and Lebeau, F. (2016). A review on the use of sensors to monitor cattle jaw movements and behavior when grazing. *Biotechnologie, Agronomie, Société et Environnement*, 20:273–286.

Andriamandroso, A., Lebeau, F., and Bindelle, J. (2015). Changes in biting characteristics recorded using the inertial measurement unit of a smartphone reflect differences in sward attributes. In *7th Conference on Precision Livestock Farming*, pages 283–289.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al. (2020). Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115.

Bahmei, B., Birmingham, E., and Arzanpour, S. (2022). CNN-RNN and data augmentation using deep convolutional generative adversarial network for environmental sound classification. *IEEE Signal Processing Letters*, 29:682–686.

Balasso, P., Marchesini, G., Ughelini, N., Serva, L., and Andrighetto, I. (2021). Machine learning to detect posture and behavior in dairy cows: Information from an accelerometer on the animal's left flank. *Animals*, 11(10):2972.

Balch, C. (1958). Observations on the act of eating in cattle. *British Journal of Nutrition*, 12(3):330–345.

Calamari, L., Soriani, N., Panella, G., Petrera, F., Minuti, A., and Trevisi, E. (2014). Rumination time around calving: An early signal to detect cows at greater risk of disease. *Journal of Dairy Science*, 97(6):3635–3647.

Chelotti, J. O., Vanrell, S. R., Galli, J. R., Giovanini, L. L., and Rufiner, H. L. (2018). A pattern recognition approach for detecting and classifying jaw movements in grazing cattle. *Computers and Electronics in Agriculture*, 145:83–91.

Chelotti, J. O., Vanrell, S. R., Milone, D. H., Utsumi, S. A., Galli, J. R., Rufiner, H. L., and Giovanini, L. L. (2016). A real-time algorithm for acoustic monitoring of ingestive behavior of grazing cattle. *Computers and Electronics in Agriculture*, 127:64–75.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN Encoder-Decoder for statistical machine translation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*. arXiv.

De Boever, J., Andries, J., De Brabander, D., Cottyn, B., and Buysse, F. (1990). Chewing activity of ruminants as a measure of physical structure—a review of factors affecting it. *Animal Feed Science and Technology*, 27(4):281–291.

Ding, L., Lv, Y., Jiang, R., Zhao, W., Li, Q., Yang, B., Yu, L., Ma, W., Gao, R., and Yu, Q. (2022). Predicting the feed intake of cattle based on jaw movement using a triaxial accelerometer. *Agriculture*, 12(7):899.

Fogarty, E. S., Swain, D. L., Cronin, G. M., Moraes, L. E., and Trotter, M. (2020). Behaviour classification of extensively grazed sheep using machine learning. *Computers and Electronics in Agriculture*, 169:105175.

Frost, A. R., Schofield, C. P., Beaulah, S. A., Mottram, T. T., Lines, J. A., and Wathes, C. M. (1997). A review of livestock monitoring and the need for integrated systems. *Computers and Electronics in Agriculture*, 17(2):139–159.

Giovanetti, V., Decandia, M., Molle, G., Acciaro, M., Mameli, M., Cabiddu, A., Cossu, R., Serra, M., Manca, C., Rassu, S., et al. (2017). Automatic classification system for grazing, ruminating and resting behaviour of dairy sheep using a tri-axial accelerometer. *Livestock Science*, 196:42–48.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., et al. (2017). CNN architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors.

Hoxhallari, K; Purcell, W. N. T. (2022). Precision livestock farming. In *10th European Conference on Precision Livestock Farming.*

Kamminga, J. W., Le, D. V., Meijers, J. P., Bisby, H., Meratnia, N., and Havinga, P. J. (2018). Robust sensor-orientation-independent feature selection for animal activity recognition on collar tags. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(1):1–27.

Khamees, A. A., Hejazi, H. D., Alshurideh, M., and Salloum, S. A. (2021). Classifying audio music genres using CNN and RNN. In Hassanien, A.-E., Chang, K.-C., and Mincong, T., editors, *Advanced Machine Learning Technologies and Applications*, pages 315–323, Cham. Springer International Publishing.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization.

Kokalis, C.-C. A., Tasakos, T., Kontargyri, V. T., Siolas, G., and Gonos, I. F. (2020). Hydrophobicity classification of composite insulators based on convolutional neural networks. *Engineering Applications of Artificial Intelligence*, 91:103613.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

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

Li, C., Tokgoz, K. K., Fukawa, M., Bartels, J., Ohashi, T., Takeda, K.-i., and Ito, H. (2021a). Data augmentation for inertial sensor data in cnns for cattle behavior classification. *IEEE Sensors Letters*, 5(11):1–4.

Li, D., Liu, J., Yang, Z., Sun, L., and Wang, Z. (2021b). Speech emotion recognition using recurrent neural networks with directional self-attention. *Expert Systems with Applications*, 173:114683.

Li, G., Xiong, Y., Du, Q., Shi, Z., and Gates, R. S. (2021c). Classifying ingestive behavior of dairy cows via automatic sound recognition. *Sensors*, 21(15).

Lim, S. J., Jang, S. J., Lim, J. Y., and Ko, J. H. (2019). Classification of snoring sound based on a recurrent neural network. *Expert Systems with Applications*, 123:237–245.

Lu, R., Duan, Z., and Zhang, C. (2018). Multi-scale recurrent neural network for sound event detection. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135.

Martinez-Rau, L. S., Chelotti, J. O., Vanrell, S. R., Galli, J. R., Utsumi, S. A., Planisich, A. M., Rufiner, H. L., and Giovanini, L. L. (2022). A robust computational approach for jaw movement detection and classification in grazing cattle using acoustic signals. *Computers and Electronics in Agriculture*, 192:106569.

Matsui, K. and Okubo, T. (1991). A method for quantification of jaw movements suitable for use on free-ranging cattle. *Applied Animal Behaviour Science*, 32(2-3):107–116.

Meng, J., Wang, X., Wang, J., Teng, X., and Xu, Y. (2022). A capsule network with pixel-based attention and BGRU for sound event detection. *Digital Signal Processing*, 123:103434.

Mesaros, A., Heittola, T., Virtanen, T., and Plumbley, M. D. (2021). Sound event detection: A tutorial. *IEEE Signal Processing Magazine*, 38(5):67–83.

Milone, D. H., Galli, J. R., Cangiano, C. A., Rufiner, H. L., and Laca, E. A. (2012). Automatic recognition of ingestive sounds of cattle based on hidden markov models. *Computers and Electronics in Agriculture*, 87:51–55.

Milone, D. H., Rufiner, H. L., Galli, J. R., Laca, E. A., and Cangiano, C. A. (2009). Computational method for segmentation and classification of ingestive sounds in sheep. *Computers and Electronics in Agriculture*, 65(2):228–237.

Monteiro, A., Santos, S., and Gonçalves, P. (2021). Precision agriculture for crop and livestock farming—brief review. *Animals*, 11(8):2345.

Nanni, L., Paci, M., Brahnam, S., and Lumini, A. (2021). Comparison of different image data augmentation approaches. *Journal of Imaging*, 7(12):254.

Navon, S., Mizrach, A., Hetzroni, A., and Ungar, E. D. (2013). Automatic recognition of jaw movements in free-ranging cattle, goats and sheep, using acoustic monitoring. *Biosystems Engineering*, 114(4):474–483.

Neethirajan, S. (2020). The role of sensors, big data and machine learning in modern animal farming. *Sensing and Bio-Sensing Research*, 29:100367.

Nydegger, F., Gyga, L., and Egli, W. (2011). Automatic measurement of jaw movements in ruminants by means of a pressure sensor. In *International Conference on Agricultural Engineering*, page 27.

Oudshoorn, F. W., Cornou, C., Hellwing, A. L. F., Hansen, H. H., Munksgaard, L., Lund, P., and Kristensen, T. (2013). Estimation of grass intake on pasture for dairy cows using tightly and loosely mounted di- and tri-axial accelerometers combined with bite count. *Computers and Electronics in Agriculture*, 99:227–235.

Papakipos, Z. and Bitton, J. (2022). Augly: Data augmentations for robustness. *arXiv preprint arXiv:2201.06494*.

Paudyal, S., Maunsell, F. P., Richeson, J. T., Risco, C. A., Donovan, D. A., and Pinedo, P. J. (2018). Rumination time and monitoring of health disorders during early lactation. *Animal*, 12(7):1484–1492.

Penning, P. D. (1983). A technique to record automatically some aspects of grazing and ruminating behaviour in sheep. *Grass and Forage Science*, 38(2):89–96.

Petmezas, G., Cheimariotis, G.-A., Stefanopoulos, L., Rocha, B., Paiva, R. P., Katsaggelos, A. K., and Maglaveras, N. (2022). Automated lung sound classification using a hybrid CNN-LSTM network and focal loss function. *Sensors*, 22(3):1232.

Ramirez, A. E., Donati, E., and Chousidis, C. (2022). A siren identification system using deep learning to aid hearing-impaired people. *Engineering Applications of Artificial Intelligence*, 114:105000.

Riaboff, L., Shalloo, L., Smeaton, A., Couvreur, S., Madouasse, A., and Keane, M. (2022). Predicting livestock behaviour using accelerometers: A systematic review of processing techniques for ruminant behaviour prediction from raw accelerometer data. *Computers and Electronics in Agriculture*, 192:106610.

Rombach, M., Südekum, K.-H., Münger, A., and Schori, F. (2019). Herbage dry matter intake estimation of grazing dairy cows based on animal, behavioral, environmental, and feed variables. *Journal of Dairy Science*, 102(4):2985–2999.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Ruuska, S., Kajava, S., Mughal, M., Zehner, N., and Mononen, J. (2016). Validation of a pressure sensor-based system for measuring eating, rumination and drinking behaviour of dairy cattle. *Applied Animal Behaviour Science*, 174:19–23.

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

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1).

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4):427–437.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Tani, Y., Yokota, Y., Yayota, M., and Ohtani, S. (2013). Automatic recognition and classification of cattle chewing activity by an acoustic monitoring method with a single-axis acceleration sensor. *Computers and Electronics in Agriculture*, 92:54–65.

Ungar, E. D., Ravid, N., Zada, T., Ben-Moshe, E., Yonatan, R., Baram, H., and Genizi, A. (2006). The implications of compound chew–bite jaw movements for bite rate in grazing cattle. *Applied Animal Behaviour Science*, 98(3-4):183–195.

Vanrell, S. R., Chelotti, J. O., Bugnon, L. A., Rufiner, H. L., Milone, D. H., Laca, E. A., and Galli, J. R. (2020). Audio recordings dataset of grazing jaw movements in dairy cattle. *Data Brief*, 30:105623.

Werner, J., Leso, L., Umstatter, C., Niederhauser, J., Kennedy, E., Geoghegan, A., Shalloo, L., Schick, M., and O'Brien, B. (2018). Evaluation of the rumiwatchsystem for measuring grazing behaviour of cows. *Journal of Neuroscience Methods*, 300:138–146.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80.

Zhu, Z., Dai, W., Hu, Y., and Li, J. (2020). Speech emotion recognition model based on bi-gru and focal loss. *Pattern Recognition Letters*, 140:358–365.

Appendix A. Computational costs

The amount of operations required for processing one second of audio signal were estimated at a sampling frequency of 6 kHz, a time window of 300 ms and a hop length of 150 ms. The procedure used to estimate these calculations is similar to the one used in Chelotti et al. (2018) in which additions and multiplications count as separated operations. The model architecture presented in Figure 4 (c) was used here for comparison purposes.

In the first block of the proposed model, the following layers were considered: re-scaling, 1D convolution and max pooling. FLOPs required for activation functions were also considered. Dropouts were discarded because these layers only applied during training, and no calculations were considered for the flatten operation. The cost of each of the convolutional layers were estimated using the following expression:

$$(2 * C_i * K * H * W * C_o) \tag{A.1}$$

where C_i and C_o represents the input and output channels, K the kernel size, H and W the size of the output feature map. According to this, the total number of FLOPs in the first block of the model is 272.235.413.

In the second block of the model, FLOPs involved in reset and update gates, activation functions and output generation were considered for every unit. The total number of FLOPs required is 191.363.413.

Finally, in the last block of the model, the FLOPs required in dense layers as well as activation functions were considered. The cost of each dense layer were estimated using the following expression:

$$(2 * I * O) \tag{A.2}$$

where I and O represent the number of input and output neurons, respectively. The total number of FLOPs in the last block of the model is 1.320.180. In summary, the total number of FLOPs in order to process one second of signal is 464.919.007.