FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

TP Integrador
2022

Inteligencia Artificial Embebida

Autor: Maximiliano Torti

Contenido

Ruta código Git

https://github.com/maxit1992/CEIA_IAE/blob/master/TP_final/TP_MaximilianoTorti.ipynb

Glosario

- *Chew – Masticación*: evento en el cual la vaca se encuentra masticando el alimento.
- *Bite – Mordida*: evento en el cual la vaca muerde y arranca la pastura del suelo.
- *Chewbite – Mordida y masticación*: evento en el cual la vaca muerde, arranca y mastica la pastura al mismo tiempo.
- *Rumination – Rumia*: actividad alimentaria que consiste en la regurgitación del material ingerido.
- *Grazing – Pastoreo*: actividad alimentaria que consiste en el arranque e ingesta de la pastura.

Sistema de detección de eventos

En este apartado, se busca crear un detector de eventos de masticación a partir de archivos de audio de duración variable que contienen eventos identificados con marcas temporales.

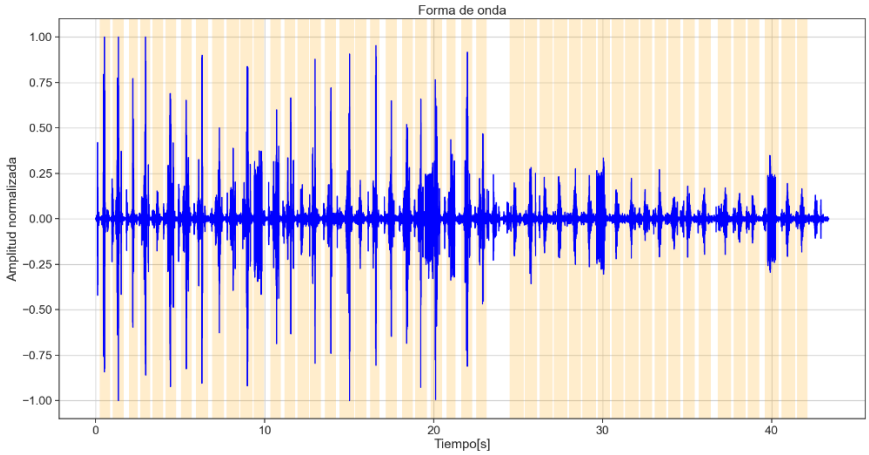

Figura 1: Ejemplo de audio con eventos masticatorios identificados

Primero se identificó que la señal de audio posee un ruido de frecuencia fija (marca de tiempo), por lo que es necesario aplicar un filtro. Al realizar la gráfica de Fourier del audio, se observó que un filtro con frecuencia de paso 2KHz y frecuencia de rechazo 2.5KHz con una atenuación de 40dB era suficiente para eliminar el ruido y mantener la información útil.

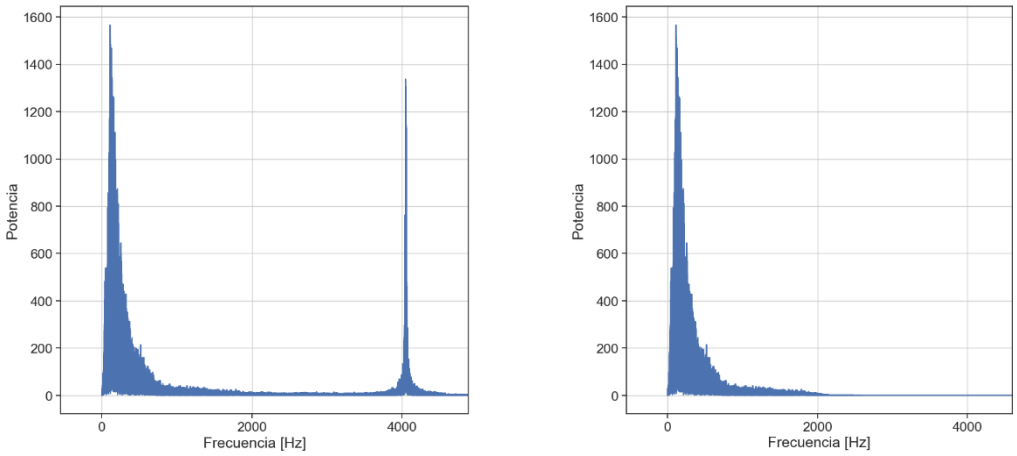

Figura 2: Gráfica frecuencial de la señal de audio antes y después del filtro

Al analizar la señal filtrada (Figura 3), se evidencia que su envolvente posee información muy útil. Se aplica entonces un filtro con frecuencia de paso 50 Hz que nos permite obtener la envolvente. Cabe destacar que este filtro genera un corrimiento, que es corregido manualmente mediante cancelación de offset.

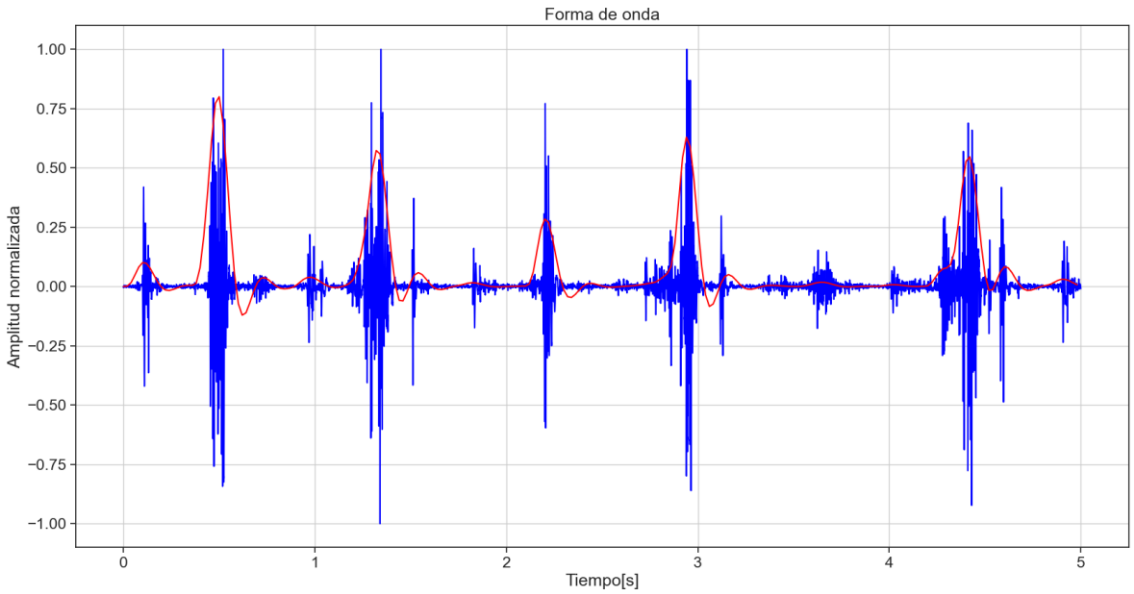

Figura 3: Señal de audio filtrada (azul) junto a su envolvente (rojo).

Sobre la señal de envolvente se probaron diferentes métodos clásicos, con guía en los resultados publicados en [1] y [2], para detección de eventos (umbrales, derivadas, energía acumulada, etc.) hasta lograr una buena performance. Los mejores resultados se lograron con los siguientes métodos:

1. **Detección de umbral**: se toman las ventanas de tiempo donde la señal envolvente supera un determinado umbral.
2. **Filtro anti-rebote**: la detección de umbral genera mucho ruido de rebote en las detecciones. El filtro anti-rebote permite unificar detecciones y tener ventanas más suaves.
3. **Expansión y limitación de ventana**: Al obtener los eventos con los métodos anteriores, se obtenía alta performance en cuanto a la cantidad de detecciones correctas, pero baja en cuanto al tiempo cubierto de los eventos (las ventanas de detección no cubrían todo el evento). Por esto se expandieron las ventanas a un tiempo mínimo y se limitaron en cuanto a su tiempo máximo.

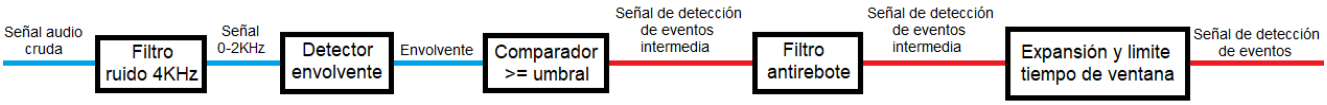

Figura 4: Resumen de esquema de procesamiento para detección de eventos.

De los bloques anteriores, surgen parámetros que deben ser fijados: umbral del comparador, ton y toff del filtro anti-rebote, y tiempo mínimo y máximo de ventana. Para obtener los parámetros, se utilizó grid-search buscando el mejor accuracy en la detección de cantidad de eventos respecto a los eventos reales proporcionados. En el caso de tiempo mínimo y máximo de ventana, como se quiere lograr buena performance respecto a la duración de los eventos, se utilizó una combinación del accuracy mencionado junto al coeficiente de dice (el cual

informa acerca de que tan bien se superponen en tiempo los eventos reales y las predicciones).

Se aclara que en el procedimiento de grid-search (podríamos llamarlo “entrenamiento”) se utilizaron solo dos audios de los tres proporcionados (“recording_01.wav” y “recording_51.wav”), mientras que el tercero se utilizó únicamente para verificación.

Los resultados obtenidos son:

	Accuracy	Coeficiente de Dice
Entrenamiento	0.879	0.772
Testeo	0.918	0.824

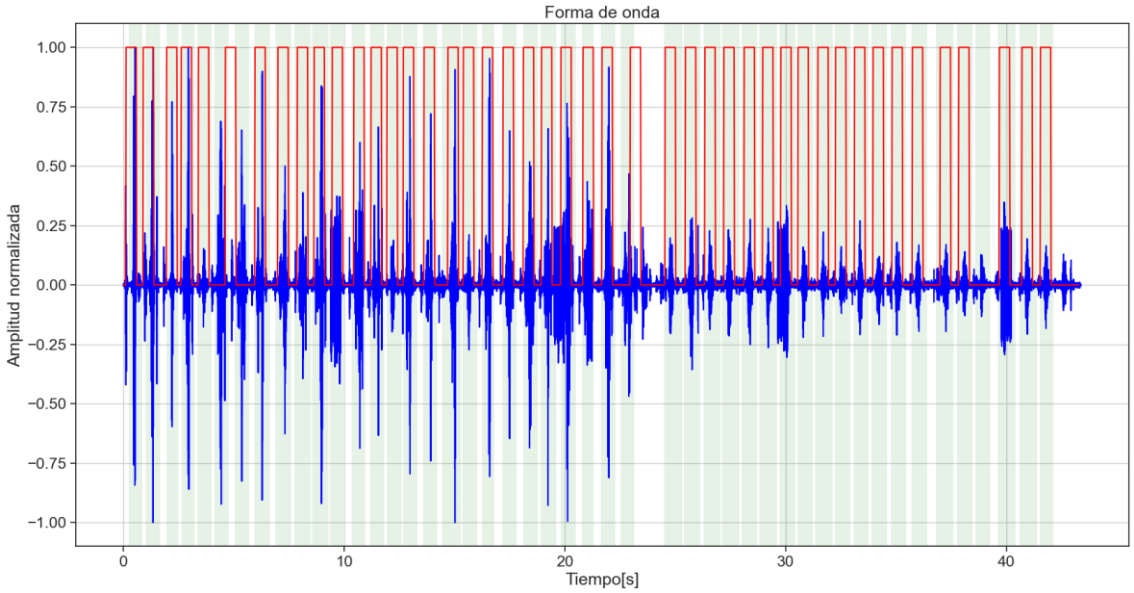

Figura 5: Señal de audio (azul) junto a los eventos reales (verde) y los detectados (rojo)

Sistema de clasificación de eventos

El clasificador de eventos se entrenó en forma separada utilizando archivos de audios de eventos recortados junto a sus etiquetas. En este caso nuevamente se aplicó el filtro anti-ruido y el filtro de envolvente.

Luego de analizar, se determina que tanto la señal filtrada como la envolvente tienen información útil, pero dada la cantidad de muestras que contienen, no es conveniente utilizarlas tal como se presentan. En su lugar, se calculan estadísticos que mantengan la información y reduzcan la carga de procesamiento del modelo de clasificación. A partir de varias pruebas y los resultados publicados en [1] y [2], los estadísticos que mejor resultado entregaron fueron:

- Duración del evento a partir de la envolvente.

- Pico máximo de la señal filtrada.
- Posición del pico máximo respecto al inicio del evento.
- Área debajo de la señal de envolvente.

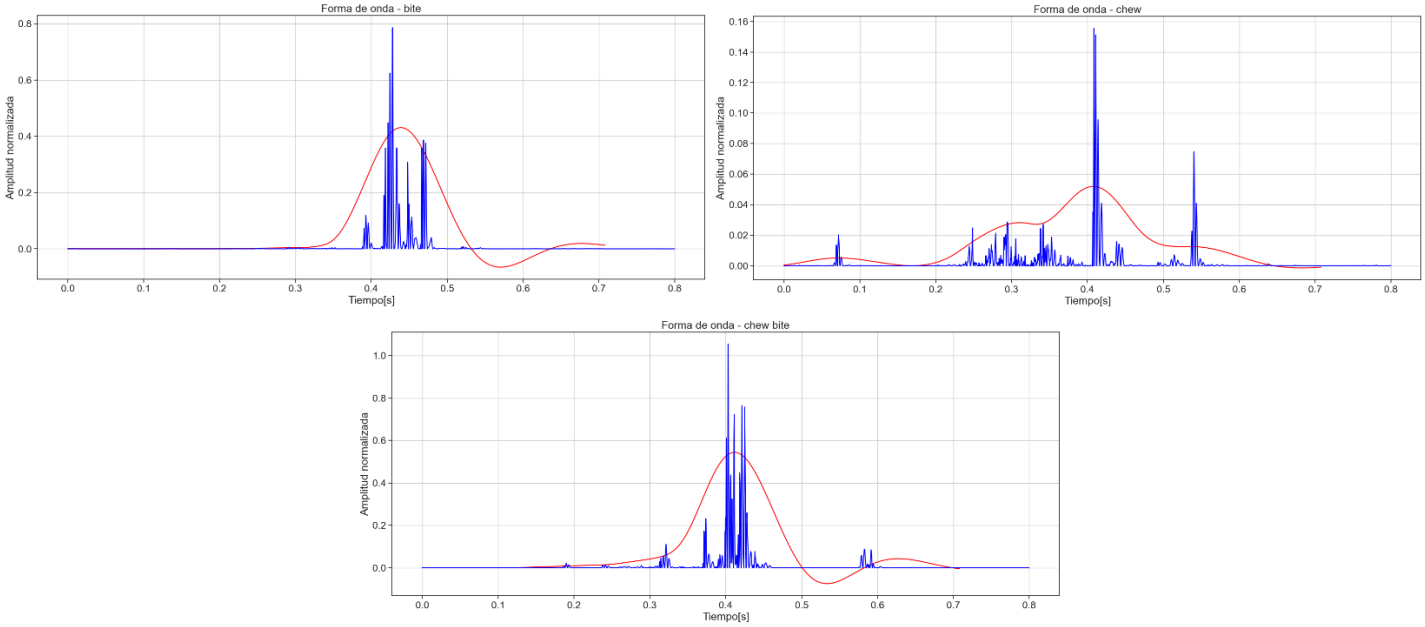

Figura 6: Señal filtrada y envolvente para los diferentes tipos de eventos

Se calcularon los estadísticos para todos los audios del Dataset, se dividió en entrenamiento y testeo (80% entrenamiento, 20% testeo) y se entrenó un modelo de árbol de decisión. Cabe aclarar que el dataset no estaba balanceado respecto a la cantidad de muestras de cada tipo de evento, lo cual generaba mala performance y overfitting al entrenar. Para solucionarlo, se aplicó undersampling sobre el dataset de entrenamiento logrando un mejor balance.

Los resultados obtenidos:

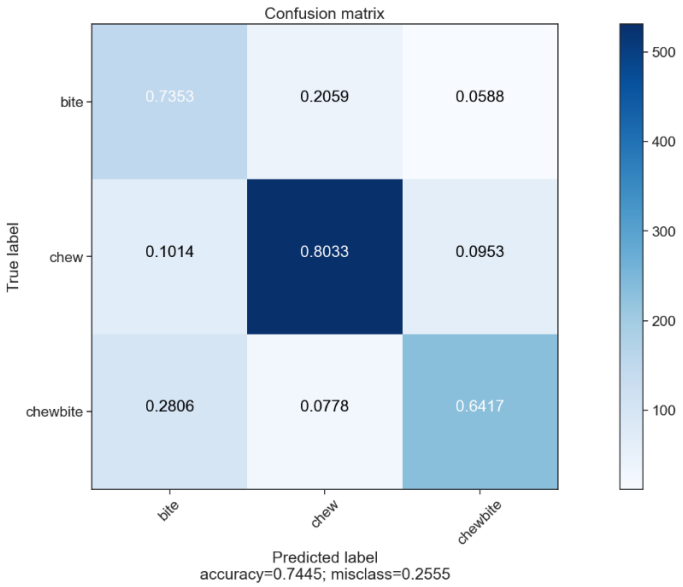

Figura 7: Matriz de confusión para dataset de entrenamiento.

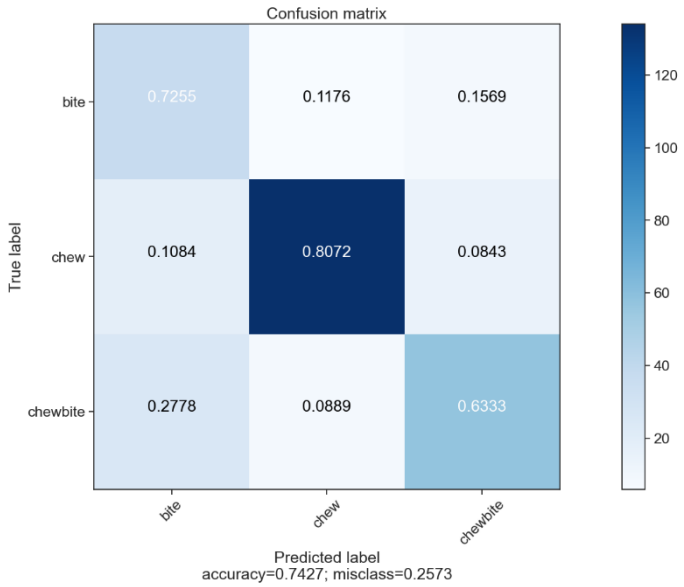

Figura 8: Matriz de confusión para dataset de testeo.

La performance obtenida es regular, con mejor precisión para “chew”. Esto es de esperarse observando que, en Figura 6, “chew” es muy diferente al resto.

A modo de integración, se unió el modelo de clasificación con el modelo de detección de eventos y se aplicó sobre los 3 audios mencionados previamente para obtener el accuracy y el coeficiente de Dice para cada tipo de evento. Los resultados obtenidos:

Clase	**recording_01.wav**		**recording_51.wav**		**recording_52.wav**	
	Accuracy	Dice	Accuracy	Dice	Accuracy	Dice
bite	0.111	0.133	0.622	0.554	0.52	0.165
chew	0.887	0.685	0.8	0.578	0.827	0.675
chewbite	0.514	0.161	0.652	0.344	0.571	0.276

Para chew los resultados son aceptables, mientras que para bite y chewbite tenemos malos resultados, en especial en el coeficiente de Dice. Esto nos indica que para estos eventos no es posible tratar la detección y clasificación de eventos en forma separada y que es necesario mejorar el modelo de clasificación.

Detección de actividad alimentaria

Para la detección de actividad alimentaria, se utilizó un archivo de audio de larga duración (6 horas) con marcas de la actividad alimentaria. El audio fue

preprocesado previamente con un software para convertirlo de formato mp3 a formato wav.

En este caso, se propone aplicar en forma continua los modelos de detección y clasificación de eventos anteriores. Una vez clasificados los eventos, se divide el tiempo continuo en bloques de 1 segundo, donde cada bloque tendrá una etiqueta correspondiente a un tipo de evento si ocurrió dicho evento o ninguna etiqueta en caso de que no se haya detectado evento. De esta manera, el stream continuo de datos de audio de alta frecuencia se reduce a un stream de 1 etiqueta por segundo. A continuación, estas etiquetas se agrupan mediante una ventana deslizante de 5 minutos de ancho y 1 segundo de desplazamiento.

Así, se convierte el audio de 6 horas en conjuntos de 5 minutos de duración con 300 etiquetas de eventos (dataset de entrada) y, por otro lado, la etiqueta de actividad para cada conjunto (dataset de salida).

Sabemos, por análisis publicado en [3], que el tipo de actividad esta muy relacionado con los eventos: en la rumia predominan los eventos de masticación; en el pastoreo predominan los eventos de mordida-masticación con porcentajes variables de eventos de mordida y de masticación, y en la ausencia de actividad alimentaria, tenemos bajos porcentajes de mordida-masticación y de masticación. Entonces, sobre los conjuntos de datos, se calcularon los siguientes estadísticos:

- Cantidad total de eventos.
- Porcentaje de eventos de masticación.
- Porcentaje de eventos de mordida.
- Porcentaje de eventos de mordida y masticación.

Armado el dataset de características, dividimos en entrenamiento y testeo y entrenamos un modelo de árbol de decisión. Los resultados obtenidos son los siguientes:

Clase	**Entrenamiento**		**Testeo**	
	Accuracy	F1-Score	Accuracy	F1-Score
Nothing	0.907	0.785	0.91	0.794
Rumination	0.97	0.964	0.969	0.963
Grazing	0.905	0.871	0.906	0.873

A pesar de que la performance del modelo de detección y clasificación de eventos no es óptima, es lo suficientemente buena como para hacer una buena clasificación de la actividad alimentaria.

Procesamiento audio crudo

Para finalizar el trabajo, se ejecuta el modelo de detección y clasificación de eventos y el modelo de clasificación de actividad alimentaria sobre un audio crudo del cual no se poseen datos de etiquetas. De estos modelos se obtienen dos archivos de salida: archivo de marcas temporales de los eventos detectados con su clasificación, y el archivo de marcas temporales de actividad alimentaria.

```
1.38    1.86    chewbite
4.16    5.12    chewbite
7.48    7.96    chewbite
8.6 9.08    chew
9.9 10.38   chewbite
10.92   11.44   chew
11.94   12.44   chewbite
16.56   17.1    bite
18.32   19.26   chewbite
```

Figura 9: Ejemplo de archivo de marcas temporales de eventos detectados con su clasificación.

```
6611    6652    Grazing
6751    6777    Grazing
6821    6832    Grazing
6840    6847    Grazing
7013    7015    Grazing
7017    7211    Grazing
7230    7271    Rumination
7274    7324    Rumination
7570    7678    Rumination
7757    7768    Rumination
7803    7806    Rumination
```

Figura 10: Ejemplo de archivo de marcas temporales de actividad alimentaria.

Para destacar, en el archivo de actividad alimentaria, notamos que la misma actividad aparece varias veces en rangos de tiempo aproximados. Estos casos deberían aparecer como un solo tramo unificado. Sería necesario cambiar los parámetros de ventana del modelo clasificador de actividad o bien realizar un post-procesamiento (por ejemplo, un anti-rebote).

Conclusiones

En general los modelos obtenidos tuvieron buena performance, lo cual implica que los preprocesamientos y las características seleccionadas son correctas.

Se observó que, mayormente, modelos simples de inteligencia artificial pueden resolver el problema, siempre que se haya hecho un buen preprocesamiento. Por esto, es fundamental el análisis y acondicionamiento de señales y la selección de características.

También se destaca que todos los pasos realizados son implementables en sistemas embebidos manteniendo un bajo consumo. Es decir que, replicando el

algoritmo de Python desarrollado en el sistema embebido, podemos construir un dispositivo de IoT con inteligencia artificial embebida para la detección y clasificación de eventos y actividades alimentarias de vacas.

Finalmente, quedan pendientes de desarrollo los siguientes puntos:

- Encontrar mejores características y mejorar la performance del modelo de clasificación de eventos para bite y chewbite.
- Mejorar la interacción entre el modelo de detección y el modelo de clasificación de eventos para obtener un mejor coeficiente de Dice para todos los tipos de eventos.
- Analizar y probar con cambios en los parámetros de ventana o realizar un post-procesamiento (una opción es un anti-rebote) en el modelo de clasificación de actividad alimentaria, de manera de tener salidas más “continuas” o “suavizadas”.

Referencias

[1] José O. Chelotti , Sebastián R. Vanrell, Diego H. Milone, Santiago A. Utsumi, Julio R. Galli, H. Leonardo Rufiner, Leonardo L. Giovanini. *A real-time algorithm for acoustic monitoring of ingestive behavior of grazing cattle*. Computers and Electronics in Agriculture (2016), 127, pp 64-75.

[2] José O. Chelotti , Sebastián R. Vanrell, Julio R. Galli, Leonardo L. Giovanini, H. Leonardo Rufiner. *A pattern recognition approach for detecting and classifying jaw movements in grazing cattle*. Computers and Electronics in Agriculture (2018), pp 83-91.

[3] José O. Chelotti, Sebastián R. Vanrell, Luciano Martinez Rau, Julio R. Galli, Alejandra M. Planisich, Santiago A. Utsumi, Diego H. Milone, Leonardo L. Giovanini, H. Leonardo Rufiner. *An online method for estimating grazing and rumination bouts using acoustic signals in grazing cattle*. Computers and Electronics in Agriculture (2020), 173, pp 105443.