Sistema Acústico De Detección De Fallos En Tiempo Real

Manuel A. Sobreira-Seoane, Eduardo Rodríguez-Calvo

[2] AtlanTTic Research Center for Telecommunications Technologies.
Universidad de Vigo. 36310

Resumen

En el presente documento se describe el proceso de desarrollo de un sistema genérico para la detección de fallos en tiempo real, a partir de la huella acústica del sistema a monitorizar. Se establece para ello un conjunto genérico de características, seleccionables, con el objetivo de minimizar los recursos computacionales necesarios para conseguir integrar el sistema en un hardware sencillo, de bajo costo. Se analizan los recursos computacionales necesarios para abordar la detección de fallos mediante dos técnicas: por una parte se evalúa la posibilidad de trabajar directamente con la representación tiempo-frecuencia de las señales y recurrir a técnicas de procesado de imagen para realizar la detección; por otro lado se evalúa el costo de un aproximación clásica de detección de eventos acústicos, recurriendo a características estadísticas, temporales y frecuenciales y a clasificadores conocidos (SVM, GMM, kNN).

Palabras clave: control de calidad fin de línea, clasificación de eventos, SVM, kNN, Machine Learning.

Abstract

In this paper, the process of developing a generic system for real time fault detection of faults in real time is describe. The system integrates the identification of the acoustic fingerprint of the noise signal emitted by the system to be monitored. A generic set of acoustic features is established with the aim of minimizing the computational resources necessary to integrate the system into a simple, low-cost hardware. The computational needs to carry the detection of faults out are analyzed. Two techniques are discussed: on the one hand, the possibility of working directly with the time-frequency representation of the signals and use image processing techniques to perform the detection is evaluated; on the other hand, the cost of a classical approach to the detection of acoustic events is evaluated, using statistical, temporal and frequency characteristics and known classifiers (SVM, GMM, kNN).

Keywords: End of Line Quality Control, Sound Event Detection, SVM, kNN.

1. Introducción

Los sistemas mecánicos constan de diversos elementos que emiten ruido de diversas características:

- ruido de flujo debido a la circulación de gases o fluidos diversos a través de conductos
- ruidos impulsivos debido a impacto
- ruidos generados por rozamiento entre componentes mecánicos
- ruidos radiados a partir de la vibración mecánica de algún elemento
- etc.

Cada uno de estas emisiones acústicas presentan un conjunto de características que constituyen su *firma o huella acústica.* Este patrón característico durante un funcionamiento normal evoluciona a lo largo del ciclo de vida de elemento mecánico hasta que en algún momento se produce un fallo. La monitorización acústica inteligente permite identificar el estado del sistema a partir de la evaluación de la huella acústica.

Cualquier cambio en la huella acústica va a tener su reflejo en alguna de las características sonoras que la constituyen. Mesaros et al. [1] realiza una descripción básica de cómo abordar el problema de detección de eventos sonoros. G. Sharma et al. [2] presenta una excelente recopilación de los métodos de extracción de características de señales de audio. Sobreira [3] utiliza parte de estas características para presentar un sistema de detección de goteos/pérdidas de agua en entornos domésticos.

En el presente artículo se describe la integración de un sistema de clasificación genérico, optimizado para que pueda funcionar en un sistema de bajo coste (Raspberry Pi 4). Como ejemplo de aplicación se muestra el funcionamiento del sistema aplicado a la detección de fallos y manipulaciones en contadores de gas (Figura 1).

Figura 1: Ejemplo de grabación de señales de audio con micrófonos de contacto para detección de fallos y manipulaciones de contadores de gas.

Para lograr un sistema de detección de eventos de audio eficiente, se evalúa un conjunto de características genéricas, seleccionando las que permiten gestionar la mayoría de los problemas asociados al mal funcionamiento de sistemas mecánicos:

- La aparición de ruidos impulsivos, por desajustes o roturas.
- Aparición de nuevas frecuencias en el espectro por fricción de elementos que giran.
- Cambio en el balance espectral por variaciones en velocidades de funcionamamiento.

2. Metodología.

2.1. La base de datos de entrenamiento

Todo sistema de inteligencia artificial/*Machine Learning* tiene que disponer de una base de datos de entrenamiento, correctamente diseñada, dimensionada y balanceada, que garantice un correcto aprendizaje del sistema. Un incorrecto diseño de la base de datos dará lugar a problemas de detección. Los problemas más habituales son:

1. Subentrenamiento o *Underfitting.* Si el conjunto de entrenamiento no cuenta con un número suficiente de muestra, el sistema funcionará bien para el conjunto de entrenamiento, pero no habrá construido unas reglas demasiado flexibles y sencillas, por lo que no funcionará *adecuadamente con el conjunto de test.*
2. *Sobreentrenamiento u Overfitting.* Alimentar el sistema con excesivos datos, o la utilización de datos desbalanceados acarrean un aprendizaje incorrecto.

Para cada posible aplicación en la que se desee aplicar el sistema desarrollado descrito en este artículo, se debe diseñar una estrategia adecuada para la generación de la base de datos: se deben seleccionar los sensores acústicos adecuados (acelerómetros, micrófonos, sensor de velocidad de partículas [5], sensor de ultrasonidos, etc) y definir una estrategia para la captación de los eventos asociados a las diversas clases que se pretenden detectar.

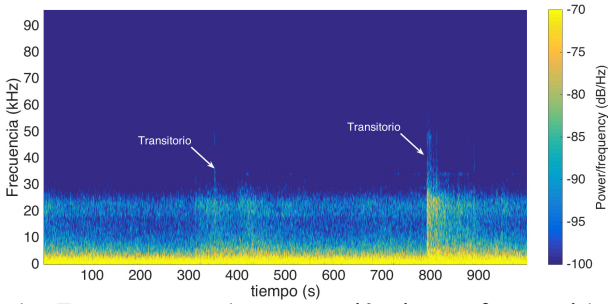
2.2. Generación del conjunto de entrenamiento: extracción de carácterísticas.

La extracción de características se puede realizar bien sobre la totalidad de un evento, o trama a trama, de forma que se puede seguir la evolución de las características de un evento a lo largo del tiempo. La estrategia de la extracción por tramas, presenta algunas características que nos han parecido interesantes a la hora de abordar un sistema de detección de eventos en tiempo real:

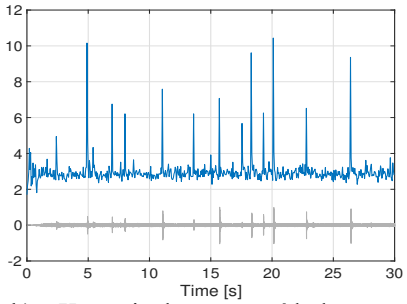
- por un lado, con tramas de duración *t* segundos, cada evento de duración I segundos genera *T/t* entradas en la base de datos
- facilita la programación en tiempo real. El sistema realizará una decisión por trama de forma contínua, decidiendo si una trama pertenece o no a una clase determinada, no siendo necesario la programación de un detector de eventos previo que permita segmentar la señal en eventos para abordar su posterior clasificación.
- se puede desarrollar alguna estrategia de postprocesado. Por ejemplo, una votación por mayoría: si de un conjunto N de tramas *n* pertenecen a la clase i, se decidirá que existe un evento perteneciente a la clase si *n* es al menos una fracción representativa del total de las tramas detectadas. De esta forma se puede mejorar la tasa de detección del clasificador, ya que en el caso de existir algúna falsa detección en alguna trama, esta quedará enmascarada por el resultado de los aciertos.

El amplio abanico de características existentes [2,7] hace necesario una selección previa de aquellas que a priori pueden presentar en general buenos resultados en muchos problemas de detección de eventos:

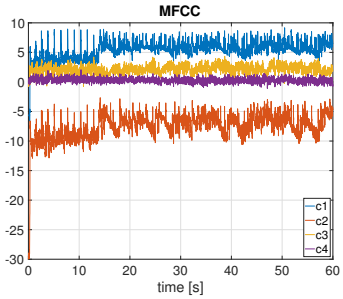
- *MFCCm*: Los *Mel-Frequency Cepstral Coeficients,* o coeficientes MEL, son un conjunto de coeficientes perceptuales, muy utilizados en reconocimiento de voz. Se obtienen a partir de un filtrado de la señal mediante un banco de filtros uniforme en escala logarítmica [6]. Una vez representada la señal se aplica la transformada del coseno discreta – DCT – para obtener el conjunto de coeficientes MEL. La utilización de estos coeficientes se trasladó al problema de detección de eventos acústicos procedentes del mundo del procesado de voz. Presentan problemas debido gran sensibilidad a la interferencia de ruido de fondo [7]. Si se opta por un cálculo de los MFCCs trama a trama, es posible observar la variación a lo largo del evento de los coeficientes (derivada primera de los MFCCs), que es más inmune al ruido de fondo en condiciones de ruido estacionario.
- *Loudness* o Sonoridad percibida. La utilización de la sonoridad percibida, calculada según se describe en la norma ISO 532:2017 [18], permite generar una característica que potencia aquellas frecuencias que son más relevantes en la percepción del evento acústico.
- *SC*: El Centroide espectral, identifica el balance de frecuencias a partir del centro de masas del espectro de la señal.
- *KS*: Curtosis, momento estadístico de cuarto orden. Permite evaluar si la distribución estadística de un conjunto de muestras es normal. La presencia de transitorios resulta en valores de kurtosis muy elevados.
- *SH*: Entropía espectral, es un buen indicador la uniformidad del espectro.



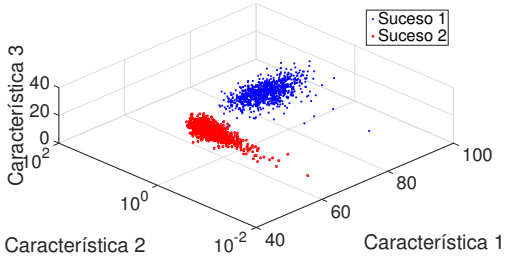
a) Espectrograma (representación tiempo-frecuencia)



b) Kurtosis durante perídodo con ruidos impulsivos.



c) MFCC



d) Scatter plot correspondientes a tres características de dos eventos de audio.

Figura 2: Ejemplo de algunas características que permiten detectar la presencia de eventos en una señal de audio.

La figura 2 representa algunos ejemplos de estas características. Además de las características mencionadas, cualquier representación bidimensional de la señal, como por ejemplo el espectrograma (variación de la magnitud del espectro de la señal a lo largo del tiempo), puede ser utilizada como imagen y utilizar para la detección de eventos de audio cualquiera de las técnicas conocidas de reconocimiento de imagen.

2.3. El etiquetado de las clases

El etiquetado debe representar de manera clara los diferentes eventos o clases que se espera que caracterice el clasificador. Puede realizarse de dos forma distintas atendiendo a su distribución en el tiempo:

- Etiquetado fuerte: Se corresponde con un etiquetado que marca el momento exacto del suceso del evento respecto al tiempo, este tipo de etiquetado se realiza mediante *timestamps* que definen la duración del evento.
- Etiquetado débil: Este tipo de etiquetado se corresponde con la aparición del suceso en la señal, independientemente de su momento de aparición, se asigna la etiqueta del suceso a la totalidad del archivo, o al *frame*, en caso de dividirse la señal.

En el caso de sistemas para detección de fallos, el carácter mutuamente excluyente de las clases (funcionamiento correto/fallo) y su persistencia en el tiempo, hace que se considere suficiente el etiquetado débil para el diseño del clasificador.

2.4. El modelo de aprendizaje (clasificador).

El conjunto de características extraídas del audio junto con las etiquetas, conforman la totalidad de datos necesarios para el entrenamiento del modelo. La naturaleza de modelo de aprendizaje dependerá del tipo de características elegidas. La elección de un sistema en el que la matriz de características se corresponda con un espectrograma como el representado en la figura 2a, con lo que una *Convolutional Neural Network – CNN –* (red neuronal convolucional) o una *Recurrent Neural Network – RNN* -- (red neuronal recurrente), son los modelos más adecuados.
En el caso de elegir como matriz de características un vector de datos, los algoritmos de aprendizaje supervisados resultan más adecuados. En el apartado 3, implementación y resultados, se comentan algunos de los métodos probados y su precisión para un ejemplo de detección de fallos y manipulaciones en contadores de gas.

2.5. Validación del modelo.

El proceso de validación se realiza seleccionando un conjunto de pruebas, diferente al conjunto de datos de entrenamiento. La comparación de las clases obtenidas de la predicción con las clases reales a las que pertenecen los eventos permite obtener marcadores de la calidad del clasificador:

- Convergencia de entrenamiento: Gráfica en la que se muestra la evolución de la sensibilidad del modelo de clasificación en función de las iteraciones de entrenamiento.
- Matriz de confusión: En ella se pueden analizar las relaciones entre verdaderos positivos (tp), falsos positivos (fp), verdaderos negativos (tn) y falsos negativos (fn) para cada clase.
- Curvas *ROC – Receiver Operation Characteristics* – Características Operacionales del Receptor: Expresan la validez de un clasificador en base al área existente bajo la curva obtenida de comparar la proporción de fp y tp para cada clase, o las medias entre las clases.
  - Macro average: media aritmética entre las clases.
  - Micro average: pondera los tp entre la suma de tp y fp
- *Accuracy o* Exactitud*:* Contabiliza los aciertos en base a los elementos totales con los que se alimenta el clasificador.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

- *Precission o* Precisión*:* Contabiliza los aciertos en base a la predicción de cada clase realizada por el clasificador.

$$Precission = \frac{TP}{TP + FP} \quad (2)$$

- Recall: Contabiliza los aciertos en base a la verdad de cada clase realizada por el clasificador.

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

- F1-Score: Media armónica del *precision* y *recall*.

$$F1 - Score = \frac{2(precission \cdot recall)}{precission + recall} \quad (4)$$

Estas medidas de calidad de los clasificadores se utilizan en el siguiente apartado para evaluar las prestaciones de los clasificadores integrados para el ejemplo de detección de fallos en contadores de gas.

3. Integración del sistema.

En este apartado se describe la integración final de un sistema de clasificación de propósito general, programado en Python, debido a la flexibilidad de trabajo que ofrece y el gran número de librerías destinadas a los procesos necesarios para el desarrollo.

3.1. Hardware

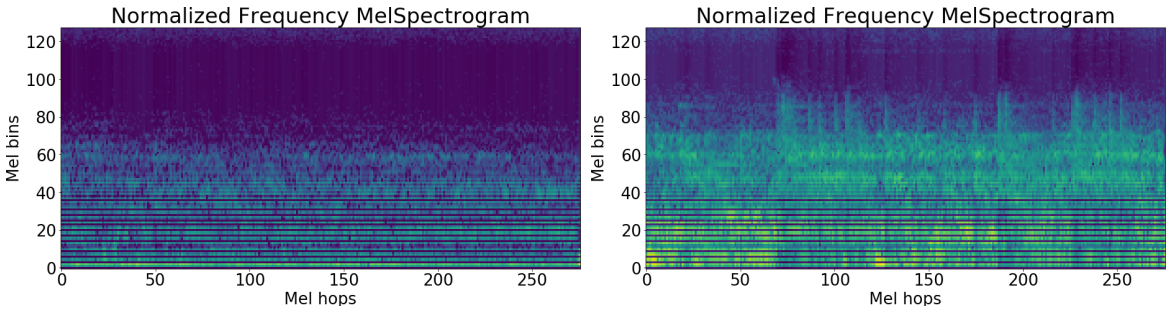
El clasificador se integra en una Raspberry Pi4B, con 8 GB de RAM. Cuenta con una pantalla táctil resistiva, y una interfaz de sonido y un micrófono de bajo coste.

3.2. Base de datos.

La base de datos de entrenamiento, consta de 8000 tramas de 1 segundo, repartidos en dos clases (funcionamiento correcto/fallo). Se realiza una validación utilizando un K-Fold estratificado con K=5, con el 80 % de los datos de la base de datos. Se realizan pruebas con distinto número de interaciones en el entrenamiento (*epochs*), cuyos resultados se presentan en el apartado 3.5.

3.3. Procesado de imagen: kapre y tensorflow.

El sistema de clasificación cuenta con la posibilidad de abordar la tarea de detección de fallos a partir de una base de datos de imágenes, con tantas clases como se estime en el problema. Se utiliza el proyecto de código abierto *audio-classification*, basado en *Kapre* [8] y *Tensorflow* [9]. *Kapre* es una librería destinada a la extracción de características de audio en tiempo real con procesamiento sobre gpu. *Tensorflow* es una librería destinada al entrenamiento de redes neuronales. La suma de estas dos tecnologías permite utilizar como matriz de características el *espectrograma MEL normalizado* [1]. La figura 3 muestra como ejemplo dos imágenes correspondientes al espectrograma MEL de contadores de gas correctos y con fallos.



(a) Funcionamineto normal (b) Fallo

Figura 3: Espectrogramas Mel en contador de gas

Configuración: El programa de entrenamiento permite configurar los siguientes parámetros de entrada:

- Frecuencia de muestreo: Configura la frecuencia de muestreo del audio de entrada al valor indicado independientemente del formato existente de la base de datos.
- Tamaño de *trama*: Indica la duración de los *Mel-spectrogram*.
- *Epochs*: Parámetro que indica el número de iteraciones del entrenamiento.

Tipo de red neuronal: Elección del modelo de entrenamiento entre, Red Neuronal Convolucional en una dimensión, Red Neuronal Convolucional en una dimensión, Red Neuronal Convolucional en dos dimensiones o el modelo de red *LSTM* [14]– *Long Short Term Memory* – que proporciona Keras.

3.4. Clasificación clásica.

3.4.1. El conjunto de características

El sistema implementado ofrece la posibilidad de trabajar con un esquema de clasificación clásica, con una aproximacion *frame by frame.* La señal de audio se divide en tramas de *m* milisegundos, y se procede a extraer un conjunto de características para a continuación alimentar (bien entrenar en la fase de entrenamieno o bien para clasificar una nueva trama), alguno de los clasificadores especificados en la tabla 1. Este acercamiento al modelo clasificador genera una matriz de características con tantas dimensiones como características seleccionadas. Surfboard [10] es una librería escrita en Python, basada en librosa, destinada a la extracción de características de audio con funcionamiento en tiempo real. El sistema permite generar un vector con todas las características integradas: las que a entender de los autores permiten abordar un gran número de problemas de detección de eventos en entornos industriales.

El vector de característicaspresenta una estructura:

$$X_n = [MFCC_i, LD, SC, SH, H, ZS] \,, \qquad (5)$$

que contiene las siguientes características de audio:

- $MFCC_i$: *Mel-Frequency Cepstral Coeficients*, son adecuados para la identificación de sonidos similares.
- *LS*: *Loudness.* Sensación de sonoridad
- *SC*: Centroide espectral, identifica el balance de frecuencias a partir del centro de masas del espectro.
- *KS*: Curtosis, indicador estadístico de cuarto orden adecuado para detectar transitorios.
- *SH*: Entropía espectral, es un buen indicador la uniformidad del espectro.
- *H*: Entropía de *Shannon*, caracteriza la uniformidad en amplitud.
- *ZS:* Tasa de cruces por cero, relacionada con la frecuencia fundamental de la señal.

Antes de alimentar el clasificador con la tabla de características, es habitual recurrir a una reduccion de las dimensiones del problema, utilizando *PCA – Principal Component Analysis* – que realiza una proyeccion del espacio de características donde la mayor varianza de cuaquier proyección de los datos recae en la primera coordenada, la segunda mayor varianza en la segunda coordenada y así

consecutivamente. De esta forma se pueden seleccionar las caracteríticas que mejor explican la variabilidad de los datos para abordar la clasificación.

3.4.2. Clasificadores.

Scikit-Learn [13] es una librería de Python destinada a la generación y análisis de modelos basados en algoritmos de aprendizaje supervisado. El sistema permite elegir entre aquellos que se estima que presentan buenos resultados en las sitaciones más habituales en la detección de eventos:

- kNN: *k vecinos más próximos.* A partir de la distancia entre el evento y los k vecinos más próximos, se asigna la clase mayoritaria del entorno.
- SVC: *Support Vector Classification.* Clasificación basada en las *Support Vector Machines:* estos clasificadores utilizan una función kernel [16] para incrementar la dimensionalidad del problema, y encontrar en el hiperespacio generarando el hiperplano que separa de forma óptima las regiones de decisión (clases). El sistema desarrollado integra SVC con *Radial Basis Function* – RFB – Kernel.
- *Random Forest Classifier.* Este clasificador utiliza el promedio del resultado de clasificación de varios clasificadores en árbol para mejorar la predicción y controlar la posibilidad de overfitting [17].
- *Naive Bayes, Linear and Quadrant Discriminants*. Conjunto de clasificacores estadísticos. Los clasificadores lineles separan dos o más clases utilizando una combinación lineal de características. El clasificador cuadrático genera una región cuadrática de decisión generada ajustando las densidades condicionales de los datos mediante la regla de Bayes.

Para la prueba del sistema se generó una base de datos de contadores de gas, con diversos fallos y manipulaciones. Dado el escaso número de ítems de alguna clase, se integró un detector biclase, etiquetando las clases como clasificador perforado o no perforado, correspondiente la clases “perforados” a los contadores deteriorados. Se entrenan diversos modelos de tipo 1 (procesado de imagen con tensor Flow y keras) y tipo 2 (clasificación directa de audio).

3.5. Tipo 1: Clasificadores basados en tensorflow y keras.

Con los parámetros de entrada disponibles se entrenan modelos convolucionales en 1 dimensión, convolucionales en 2 dimensiones y *lstm* con los siguientes parámetros:

- fs: 16 kHz, *frames*: 1 segundo, *epochs*: 20.
- fs: 16 kHz, *frames*: 1 segundo, *epochs*: 10.
- fs: 16 kHz, *frames*: 2 segundos, *epochs*: 10.
- fs: 44,1 kHz, *frames*: 1 segundo, *epochs*: 10.
- fs: 44,1 kHz, *frames*: 2 segundos, *epochs*: 10.

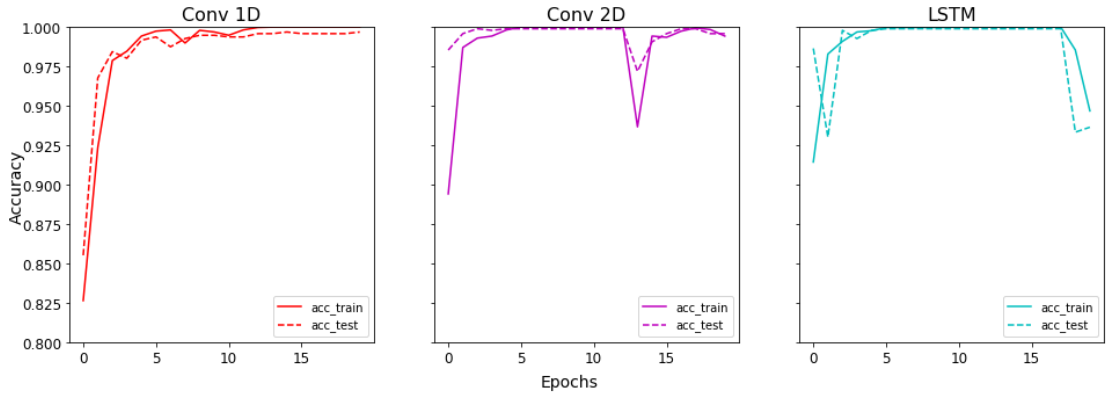


Figura 4: *Accuracy* en sistema de clasificación con Tensorflow y keras. Entrenamiento fs: 16 kHz, *duración de tramas*: 1 segundo, *epochs*: 20

4. Prueba del sistema: detección de fallos y manipulaciones en contadores de gas.

La figura 4 muestra el *accuracy* para un sistema entrenado con 20 epochs, tasa de muestreo a 16 kHz y 1 segundo de duración para cada trama. En esta configuración se obtiene una accuracy próxima al 100 % .

4.1. Tipo 2: Clasificación de audio clásica.

La tabla 1 presenta los resultados obtenidos (accuracy) para las distintas opciones integradas en el sistema de clasificación. Se observa como kNN, SVC

con RBF Kernel y el Random Forest presentan un accuracy superior al 99 %.

Clasificación Clásica: Accuracy	
Clasificador	*Accuracy*
KNeighbors Classifier	99.27 %
SVC	92.13 %
SVC RBF kernel	99.28 %
Decision Tree Classifier	96.38 %
Random Forest Classifier	99.17 %
AdaBoost Classifier	95.34 %
Gaussian NB	80.43 %
Quadratic Discriminant Analysis	89.54 %

Tabla 1. Accuracy en la detección de contadores de gas defectuosos.

4.2. Evaluación de los recursos computacionales necesarios.

Los resultados de detección son óptimos tanto los clasificadores de tipo 1 como para los de tipo 2. La decisión de integración se basará por tanto en la utilización de los mínimos recursos posibles. La caracterización del aprovechamiento de recursos se realiza midiendo en el equipo de desarrollo[1] el tiempo que tarda el programa en extraer las características y clasificarlas por cada iteración de un *frame* de un segundo. Se mide también la reserva de memoria ram por cada iteración. En la tabla 2 se pueden comprobar los resultados obtenidos para diferentes modelos. El mejor aprovechamiento de recursos se obtiene para los modelos de menor frecuencia de muestreo y para los modelos Tipo 2.Los tiempos de procesado no se consideran determinantes en la elección del modelo, ya que son pequeños respecto al tamaños del *frame*, esta relación indica que será posible realizar el proceso de clasificación de un *frame* mientras se almacena el siguiente en el buffer. El consumo de RAM tampoco se considera un factor determinante, ya que los consumos son bajos en comparación con las memorias disponibles. Aún no siendo determinantes se consideran mejores cuanto más bajos.

Recursos computacionales		
Clasificador	*Extracción+Test (ms)*	*Uso RAM (MB)*
Tipo 1 - 16 kHz	185	3,5
Tipo 1 - 44,1 kHz	220	3,5
Tipo 2 - 16 kHz	91	0,033
Tipo 2 - 44,1 kHz	117	0,088

Tabla 2. Comparación de aprovechamiento de recursos entre modelos clasificadores. Unidades por iteración de procesado de un *frame* de 1 segundo.

[1] Especificaciones equipo de desarrollo: i5-7300U CPU @ 2.60GHz, GPU HD Graphics 620 y 8GB de memoria RAM.

5. Conclusiones.

En este artículo se ha descrito la integración de un clasificador genérico de eventos de audio, aplicable a la detección de fallos en sistemas mecánicos. Se ha evaluado con una base de datos correspondiente a contadores de gas clasificados en dos clases: perforados (deteriorados o manipulados) y no perforados (funcionamiento correcto).

El análisis realizado concluye que es posible integrar un sistema de clasificación en un hardware muy sencillo (Raspberry Pi4B, con 8 GB de RAM). De las opciones analizadas, en el hardware seleccionado es posible implementar en tiempo real tanto un clasificador basado en Tensor Flow y Keras, que procesa las imágenes generadas a partir de la representación tiempo-frecuencia de la señal acústica, como un clasificador tradicional, en el que se extraen un conjunto de características de audio para a continuación pasarlas a algún tipo de clasificador. Las pruebas de funcionamiento del prototipo se han realizado emulando condiciones reales de medición, tanto en laboratorio como en entornos diversos con distintas características acústicas.

El sistema puede ser entrenado para cualquier problema de detección de eventos de audio, tanto para dos clases como múltiples clases. Se ha probado para diversas aplicaciones, presentando en todos los casos resultados muy aceptables.

El sistema consta de:

- Una raspberry Pi4B, con 8 GB de RAM
- Una pantalla táctil resistiva, ADAFRUIT INDUSTRIES PiTFT Plus de 3.5 pulgadas
- Interfaz sonido Ugreen.
- Micrófono Sennheiser XS Lav Mobile.
- Soporte de Batería 18650 con Indicador LED.
- Batería SONY VTC5A
- Cargador I4 Smart GOLISI.

6. Referencias

[1] A. Mesaros, T. Heittola, T. Virtanen, and M. D. Plumbley, “Sound event detection: A tutorial,” IEEE Signal Processing Magazine, vol. 38, no. 5, pp. 67–83, 2021.

[2] G. Sharma, K. Umapathy, and S. Krishnan, “Trends in audio signal feature extraction methods,” *Applied Acoustics*, vol. 158, p. 107020, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003682X19308795

[3] M. Sobreira, “Drip detection through its acoustic signature with variable background noise”. Proceedings Internoise 2019.

[4] Fabris, F., Freitas, A.A. “Analysing the Overfit of the Auto-sklearn Automated Machine Learning Tool”. In: Nicosia, G., Pardalos, P., Umeton, R., Giuffrida, G., Sciacca, V. (eds) Machine Learning, Optimization, and Data Science. LOD 2019. Lecture Notes in Computer Science(), vol 11943. Springer, Cham.
Doi: https://doi.org/10.1007/978-3-030-37599-7_42

[5] Hans Elias de Bree and Finn Jacobsen. “The microflon particle velocity sensor”. In Havelock, D., Kuwano, S., Vorländer, M. (eds) Handbook of Signal Processing in Acoustics. Springer, New York, NY.
Doi: https://doi.org/10.1007/978-0-387-30441-0_68

[6] Vozáriková, E., Juhár, J., Čižmár, A. (2011). Acoustic Events Detection Using MFCC and MPEG-7 Descriptors. In: Dziech, A., Czyżewski, A. (eds) Multimedia Communications, Services and Security. MCSS 2011. Communications in Computer and Information Science, vol 149. Springer, Berlin, Heidelberg.
Doi: https://doi.org/10.1007/978-3-642-21512-4_23

[7] Xiaodan Zhuang, Xi Zhou, T. S. Huang and M. Hasegawa-Johnson, "Feature analysis and selection for acoustic event detection," 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, 2008, pp. 17-20,

[8] Doi:https://doi.org/10.1109/ICASSP.2008.4517535

[9] K. Choi, D. Joo, and J. Kim, “Kapre: On-gpu audio preprocessing layers for a quick implementation of deep neural network models with keras,” in Machine Learning for Music Discovery Workshop at 34th International Conference on Machine Learning. ICML, 2017.

[10] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “Tensorflow: A system for large-scale machine learning,” in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), 2016, pp. 265–283.

[11] R. Lenain, J. Weston, A. Shivkumar, and E. Fristed, “Surfboard: Audio feature extraction for modern machine learning,” 2020.

[12] Available on line: https://pandas.pydata.org/docs/.

[13] Available:https://docs.python.org/3/library/tkinter.html

[14] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, “API design for machine learning software: experiences from the scikit-learn project,” in ECML PKDD Workshop: Languages for Data Mining and Machine Learning, 2013, pp. 108–122.

[15] A. Nanduri and L. Sherry, "Anomaly detection in aircraft data using Recurrent Neural Networks (RNN)," 2016 Integrated Communications Navigation and Surveillance (ICNS), 2016, pp. 5C2-1-5C2-8,
Doi:https://doi.org/10.1109/ICNSURV.2016.7486356

[16] Thomas Hofmann. Bernhard Schölkopf. Alexander J. Smola. "Kernel methods in machine learning." Ann. Statist. 36 (3) 1171 - 1220, June 2008.
Doi:https://doi.org/10.1214/009053607000000677

[17] Shaik, A.B., Srinivasan, S. (2019). A Brief Survey on Random Forest Ensembles in Classification Model. In: Bhattacharyya, S., Hassanien, A., Gupta, D., Khanna, A., Pan, I. (eds) International Conference on Innovative Computing and Communications. Lecture Notes in Networks and Systems, vol 56. Springer, Singapore.
Doi: https://doi.org/10.1007/978-981-13-2354-6_27

[18] ISO 532:2017. Acoustics—Methods for calculatinc Loudness.