UNIVERSIDADE DA CORUÑA

Aprendizaje Automático

MEMORIA

Sistema de reconocimiento de sonidos de mobs de Minecraft

José Manuel Amestoy López
Diego Suárez Ramos
Jordi Núñez Arias
Pablo Fernández Pérez
Eva Maria Arce Ale

Contents

1 Introducción

Los videojuegos han tenido un gran impacto en la sociedad actual, ya que han evolucionado de ser un simple entretenimiento a convertirse en una industria multimillonaria. Además de proporcionar diversión, los videojuegos también han demostrado tener beneficios educativos, psicológicos y sociales. Algunos argumentan que los videojuegos pueden mejorar la coordinación, la memoria y la resolución de problemas, mientras que otros señalan que pueden fomentar la creatividad, el trabajo en equipo y la inclusión social.

En este trabajo nos centraremos en Minecraft, un juego de mundo abierto que ha ganado una gran popularidad en todo el mundo. En este juego, los jugadores interactúan con un mundo virtual, construyendo y creando estructuras, explorando diferentes terrenos y luchando e interactuando con una gran variedad de mobs o criaturas. Cada mob tiene sus propias características y, por lo tanto, produce sonidos únicos. El objetivo de este trabajo de aprendizaje automático es desarrollar un sistema capaz de clasificar los sonidos de los diferentes mobs de Minecraft. Para ello, vamos a utilizar cuatro técnicas de aprendizaje automático para clasificar los audios: RR.NN.AA. (Redes de Neuronas Artificiales), SVM (Máquinas de Soporte Vectorial), Árboles de Decisión y KNN (K-Nearest-Neighbor o K Vecinos Cercanos)

La clasificación precisa de los sonidos de mobs es una tarea desafiante debido a la variabilidad inherente en la forma en que se producen los sonidos, por eso inicialmente el sistema será capaz de clasificar un número reducido de mobs y con el tiempo se irá incrementando este número. Además, el desarrollo de un sistema de clasificación efectivo podría ser de gran utilidad para personas con alguna discapacidad auditiva, permitiéndoles percatarse de la presencia de posibles enemigos. También podría ser interesante integrarlo con otros sistemas que por ejemplo, tengan como objetivo aprender a jugar a Minecraft. Aunque en este trabajo nos centraremos en un videojuego concreto, la expansión de este sistema a otros juegos podría ser un paso adelante para hacerlos más accesibles para muchas personas, permitiendo además otras mejoras en los mismos, como un comportamiento más realista de los personajes no jugables.

Empezaremos describiendo el problema detalladamente en el siguiente apartado 2 para luego explicar las diferentes aproximaciones al problema. Se mostrarán los resultados de estas aproximaciones en el apartado 4 haciendo uso de imágenes y tablas con los resultados obtenidos y se incluirá una discusión en la que se indicará la técnica que nos ha proporcionado los mejores resultados. También realizaremos un análisis de la bibliografía utilizada en el apartado 8, con un apartado donde consultarla. Terminaremos con unas conclusiones finales y la solución al problema propuesto en el apartado 6.

2 Descripción del problema

El objetivo es entrenar al sistema para que pueda identificar de manera precisa y eficiente los diferentes tipos de sonidos que emiten los mobs de Minecraft. Los sonidos de los mobs de Minecraft pueden ser muy variados y se utilizan para indicar una amplia variedad de situaciones y eventos. Por ejemplo, algunos mobs emiten sonidos cuando detectan al jugador, cuando son atacados, cuando mueren, etc. Además, algunos mobs emiten sonidos únicos que los identifican específicamente, como el sonido característico de los zombies o los esqueletos.

El sistema de clasificación automático deberá ser capaz de reconocer y distinguir estos sonidos, y asignarlos a las categorías correctas. Para lograr esto, será necesario recolectar una gran cantidad de datos de sonidos de mobs de Minecraft, etiquetarlos correctamente y utilizar técnicas de aprendizaje automático para entrenar al sistema en la tarea de clasificación de sonidos.

Como hemos mencionado anteriormente vamos a utilizar cuatro técnicas de aprendizaje automático para clasificar audios. Para comparar los resultados de cada técnica, vamos a utilizar dos métricas importantes: la precisión media de clasificación y el valor F1 medio.

Existen algunas restricciones que deben ser consideradas al resolver el problema de clasificación automática de los sonidos de los mobs de Minecraft. La primera está relacionada con la calidad del sonido: Para que el sistema de clasificación automático pueda identificar correctamente los diferentes tipos de sonidos de los mobs, es importante que los sonidos sean de alta calidad y limpios. Si los sonidos están distorsionados o tienen ruido de fondo, podría ser difícil para el sistema identificarlos correctamente. En segundo lugar, también hay que tener en cuenta la variabilidad en los sonidos: Los sonidos emitidos por los mobs pueden variar dependiendo del contexto en el que se encuentren. Por ejemplo, el sonido emitido por un zombie cuando persigue al jugador es diferente del sonido emitido cuando está siendo atacado. El sistema de clasificación automático debe tener en cuenta estas variaciones en los sonidos.

La BD (Base de Datos) consta de 520 audios de 104 mobs diferentes de Minecraft (cada mob tiene unos 5 audios). Estas señales son de una duración reducida (sobre 1 segundo). A nivel de formato todos los archivos son .ogg con 44.1 kHz de muestreo. Especificaremos más adelante las características que se extraerán de las señales para su futura clasificación en cada una de las aproximaciones.

Con respecto al origen, usamos como base de datos los propios archivos del juego, descargados de internet, concretamente los de la versión 1.17. Esto implica que los sonidos no incluyen ningún tipo de ruido. Fuente: Mine-imator forums[1].

3 Análisis bibliográfico / Estado del arte

El reconocimiento de sonidos es una tarea fundamental en el campo de la inteligencia artificial y el aprendizaje automático, y se ha aplicado en diversos ámbitos. A pesar de la falta de información relacionada con este tema en el ámbito de los videojuegos, existe un trabajo previo [2] que relaciona sonidos y videojuegos (aunque los sonidos que se reconocen no son del propio videojuego). El objetivo de este trabajo es explicar cómo crear software que pueda identificar las notas musicales tocadas por un instrumento en la interfaz de un juego de ordenador. Debido a la robustez del algoritmo SWIPEP se pudo reconocer exitosamente la altura musical de todas las notas de la flauta dulce soprano, tocadas por el jugador en los experimentos realizados. Con el equipo empleado en el experimento, se alcanzó medir que la implementación en C permite analizar grabaciones de más de 100 segundos , contrario a la implementación en Matlab.

Además, hay otros trabajos similares al reconocimiento de sonidos que realiza nuestro sistema. Por ejemplo, el estudio [3] publicado en Scientific Reports propone un método para clasificar automáticamente los sonidos de las aves a partir de grabaciones de audio sin procesar utilizando redes neuronales convolucionales. Asimismo, el estudio [4] publicado en Applied Sciences presenta una técnica para clasificar los sonidos de los animales utilizando espacios de disimilitud, que son representaciones basadas en la distancia entre los sonidos y un conjunto de referencias. Estos trabajos demuestran el potencial y la diversidad de las aplicaciones del reconocimiento de sonidos en el ámbito de la bioacústica. Por otro lado, el estudio [5] revisa los métodos y aplicaciones de la detección de sonidos anómalos (ASD) con aprendizaje automático, que es la tarea de identificar si el sonido emitido por un objeto es normal o anómalo. Asimismo, otro trabajo [6] presenta un sistema basado en redes neuronales convolucionales (CNN) para reconocer el canto de diferentes especies de aves a partir de grabaciones de audio. También se encuentra el trabajo [7] que presenta dos modelos de red neuronal convolucional (CNN) para resolver un problema de clasificación de sonidos ambientales en siete categorías distintas. Otro estudio [8] propone usar descriptores de audio basados en MPEG-7 para el reconocimiento de voces de animales en redes de sensores acústicos inalámbricos. Los autores comparan el rendimiento de diferentes descriptores de audio para clasificar las señales sonoras de cuatro especies animales: ranas, aves, perros y vacas. Por último, el estudio [9] presenta un proyecto que ayuda a clasificar los ruidos urbanos en etiquetas usando diferentes modelos de aprendizaje automático y redes neuronales, como SVM, MLP y CNN, con el objetivo de minimizar, predecir y analizar el sonido urbano para mejorar la calidad de vida.

En el presente trabajo, no nos basaremos en ninguno de los mencionados en la bibliografía debido a que ninguno se adecua de manera óptima a nuestro problema concreto. Por lo tanto, vamos a abordar el desarrollo de nuestro trabajo a través de diversas aproximaciones que iremos detallando a continuación. Es importante destacar que esta metodología nos permitirá diseñar una solución personalizada y específica para nuestro problema en particular, asegurándonos de que se adapte perfectamente a las necesidades y requisitos que se presenten.

4 Desarrollo

Realizaremos varias aproximaciones para garantizar el correcto desarrollo del sistema. Empezaremos con una versión simplificada del problema original.

4.1 Aproximación 1 :

4.1.1 Descripción:

En esta primera aproximación nos centraremos en distinguir entre zombie y no zombie. Los audios que obtenemos de la BD están en formato ".ogg", pero los pasamos a formato".wav" para facilitar la fragmentación de cada audio. Utilizamos concretamente 6 audios de zombie, 11 de zombie piglin, 6 de vaca, 3 de oveja, 16 de aldeano, 5 de araña. De estos audios obtuvimos 45 fragmentos de audio de zombies (zombie y zombie piglin), y 47 fragmentos de audio de no zombies (vaca, araña, aldeano y oveja). Podemos ver las gráficas de uno de estos fragmentos de audio en la Figura 1. Para realizar esta fragmentación, como la frecuencia de los audios es de 44100 muestras por segundo, empleamos un tamaño de ventana de 2^{15} muestras (32768 muestras, aproximadamente medio segundo) y además establecemos un porcentaje de solapamiento de 0.5. De cada ventana extraemos las características con la FFT(Transformada de Fourier). Inicialmente tendremos dos características para cada patrón, que serán la media y desviación típica de cada fragmento de audio. Normalizamos los datos con media y desviación típica porque estamos trabajando con frecuencias y pueden tener valores muy altos y muy bajos. La normalización por mínimo y máximo es sensible a los valores atípicos, es decir, si tenemos valores extremos que difieren significativamente del resto de los valores, esta normalización podría distorsionar el rango de valores, es por esto que creemos que no es una buena opción. Para cada patrón debemos determinar si pertenece o no a un zombie. El número de folds empleado será 10 en todos los casos.

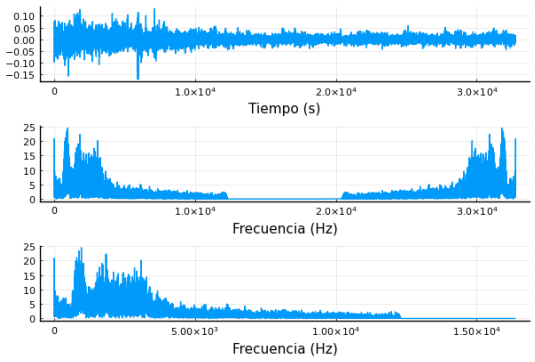

Figure 1: grafica

4.1.2 Resultados:

En primer lugar, en el proceso de entrenamiento de nuestra Red de Neuronas Artificiales (RR.NN.AA.) para clasificar audios de Minecraft como zombie o no zombie, hemos evaluado 8 configuraciones distintas. Todas las configuraciones tienen 2 neuronas de entrada y 1 de salida, y varían en la cantidad y la disposición de las capas ocultas. Las configuraciones incluyen una capa oculta con 5, 10, 15 o 20 neuronas, así como varias opciones de dos capas ocultas, cada una con una combinación diferente de neuronas. Hemos seleccionado estas configuraciones porque nuestro problema concreto tiene pocos patrones y pocas características, por lo que hemos creído que una red con un número limitado de neuronas en las capas ocultas sería suficiente para obtener un buen rendimiento de clasificación. A través de la evaluación de estas configuraciones, buscamos encontrar la que obtenga el mejor resultado en términos de precisión de clasificación. En todos los casos, hemos establecido el número de folds a 10.

En la tabla 1 se muestra un resumen de los resultados obtenidos para cada configuración. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

Configuración	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
2-5-1	68.26	16.01	63.14	22.80	0.11	1.00	0.80
2-10-1	68.96	16.16	64.56	21.99	0.11	0.86	1.00
2-15-1	68.80	15.49	64.59	20.98	0.11	1.00	0.80
2-20-1	68.56	15.35	64.36	20.80	0.11	1.00	0.80
2-5-15-1	68.22	13.94	66.06	17.09	0.11	1.00	0.80
2-10-15-1	68.12	14.81	66.22	17.20	0.11	1.00	0.80
2-15-15-1	67.82	14.54	64.96	18.14	0.11	1.00	0.80
2-15-20-1	67.41	14.30	64.84	17.50	0.11	1.00	0.80

Table 1: Tabla de resultados RR.NN.AA. 1

En la tabla 2 se muestra la matriz de confusión correspondiente a la configuración 2-5-15-1 (20 neuronas en una sola capa oculta), que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	4	1	28	14
Zombie	0	4	13	28

Table 2: Matriz de Confusión de la 7ª iteración (Configuración : 2-5-15-1)

En el proceso de entrenamiento de SVM hemos utilizado 8 configuraciones distintas, variando los parámetros de kernel, gamma y C. Hemos utilizado el kernel "rbf", ya que es uno de los más utilizados en problemas de clasificación y es adecuado para datos no linealmente separables. Hemos probado con distintos valores de gamma y C, que son hiperparámetros que influyen en la complejidad del modelo y en su capacidad para generalizar. En nuestro problema concreto, con pocos patrones y pocas características, hemos elegido valores de C relativamente altos y gamma bajos o medios para evitar el sobreajuste y mejorar la capacidad de generalización del modelo. A través de esta exploración de hiperparámetros, hemos tratado de encontrar la configuración óptima para nuestro problema de clasificación de sonidos. En todos los casos, hemos establecido el número de folds a 10.

En la tabla 3 se muestra un resumen de los resultados obtenidos para cada configuración. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

kernel	**gamma**	**C**	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
rbf	0.01	100	69,77	11.44	64,26	17.07	0.11	1.00	0.80
rbf	0.01	1000	72,99	11.23	68.12	15.82	0.89	0.14	0.00
rbf	0.01	10000	73.88	14.008	70.96	14.95	0.89	0.17	0.00
rbf	0.001	10000	73.00	11.44	69.43	16.23	0.11	1.00	0.90
rbf	0.01	10000	69.66	14.21	62.20	19.62	0.89	0.14	0.00
rbf	0.0001	1000	64.22	13.28	63.6	13.16	0.11	1.00	0.80
rbf	0.001	100000	71.89	12.42	69.10	14.25	0.11	1.00	0.80
rbf	0.0001	100000	72.89	18.76	68.46	21.86	0.89	0.17	0.00

Table 3: Tabla de resultados SVM 1

En la tabla 4 se muestra la matriz de confusión correspondiente a la tercera configuración (kernel: "rbf", gamma: 0.01, C: 10000), que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	1	1	4	38
Zombie	0	4	28	13

Table 4: Matriz de Confusión de la 7ª Iteración (kernel: "rbf", gamma: 0.01, C: 10000)

Para el modelo de árboles de decisión, hemos evaluado seis configuraciones distintas variando la profundidad máxima del árbol desde 1 hasta 6. Creemos que este rango de valores es adecuado para nuestro problema, dado que contamos con un conjunto de datos de tamaño moderado, con 92 patrones y 2 características cada uno. Al probar con diferentes valores de profundidad, hemos buscado encontrar un balance entre la complejidad del modelo y su capacidad para capturar patrones en los datos, evitando tanto el sobreajuste como el subajuste. En todos los casos, hemos establecido el número de folds a 10.

En la tabla 5 se muestra un resumen de los resultados obtenidos para cada configuración. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

depth	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
1	69.44	12.78	73.56	10.00	0.11	1.00	0.8
2	70.55	11.89	74.99	9.66	0.11	1.00	0.8
3	64.00	9.24	54.05	11.54	0.76	0.49	0.00
4	78.44	9.76	78.66	11.61	0.11	1.00	0.67
5	73.00	8.45	69.08	14.13	0.11	1.00	0.67
6	69.67	8.05	67.34	9.71	0.22	0.67	1.00

Table 5: Tabla de resultados

En la tabla 6 se muestra la matriz de confusión obtenida con profundidad 4, que es la mejor opción para nuestro problema dado que alcanzó la mayor precisión media de clasificación y valor F1 medio.

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	2	1	33	11
Zombie	0	6	3	33

Table 6: Matriz de Confusión de la 4ª Iteración (Profundidad 4)

Para el algoritmo K-Nearest Neighbors (KNN) hemos utilizado seis configuraciones diferentes para el número de vecinos, de 1 a 6. Creemos que estas son opciones adecuadas dado que tenemos un conjunto de datos con pocos patrones y pocas características. Al utilizar un número bajo de vecinos, estamos limitando el espacio de búsqueda y evitando el sobreajuste del modelo, de esta forma, KNN es capaz de generalizar mejor y producir modelos más precisos para este tipo de datos. En todos los casos, hemos establecido el número de folds a 10.

En la tabla 7 se muestra un resumen de los resultados obtenidos para cada configuración. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

k	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
1	66.44	13.53	60.44	14.94	0.89	0.14	0.00
2	59.89	12.90	42.01	21.32	0.78	0.33	0.17
3	66.44	11.32	61.50	15.71	0.78	0.29	0.00
4	62.11	14.89	50.43	21.76	0.78	0.29	0.00
5	66.44	14.50	62.47	18.24	0.11	1.00	0.80
6	60.89	12.63	53.73	15.98	0.78	0.25	0.20

Table 7: Tabla de resultados KNN 1

En la tabla 8 se muestra la matriz de confusión obtenida con k=5, dado que se alcanzó la mayor precisión media de clasificación y valor F1 medio.

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	4	1	9	33
Zombie	0	4	28	13

Table 8: Matriz de Confusión de la 7ª Iteración (k = 5)

4.1.3 Discusión:

Después de analizar los resultados obtenidos, llegamos a la conclusión de que la técnica que nos ha proporcionado los mejores resultados ha sido la de Árboles de Decisión, concretamente para una profundidad máxima de 4, dado que se alcanzó la mayor precisión media de clasificación y valor F1 medio, con un error de entrenamiento de 0.17 y un error de test de 0.11, que son valores aceptables.

Para la siguiente aproximación todavía tenemos bastante margen de mejora con los datos actuales, creemos que lo mejor sería no complicar la BD y añadir nuevas características que capten más información, puesto que es posible que las que tenemos actualmente sean demasiado básicas. De esta forma, es probable que el modelo tenga una mayor capacidad para diferenciar entre los diferentes tipos de sonidos y, por lo tanto, se obtengan mejores resultados en la clasificación con las distintas técnicas.

En la tabla 9 se muestra un resumen de los mejores resultados de cada modelo

Técnica	**Average test accuracy**	**Average test F1**	**Configuración**
RR.NN.AA.	68.22	66.06	2-5-15-1
SVM	73.88	70.96	kernel: "rbf", gamma: 0.01, C: 10000
ARBOLES	78.44	78.66	Profundidad: 4
KNN	66.44	62.47	K = 5

Table 9: Tabla de los mejores resultados de cada técnica

4.2 Aproximación 2 :

4.2.1 Descripción:

Hasta ahora sólo habíamos utilizado media y desviación típica de toda la señal en frecuencia. En esta segunda aproximación, hemos añadido dos nuevas características, la media y la desviación típica de un trozo de la frecuencia, concretamente desde los 1345.8 Hz (para descartar los valores más bajos) hasta los 6858.32Hz. Hemos seleccionado este fragmento en base a la observación de las gráficas, llegando a la conclusión de que es la parte en la que más se aprecian las diferencias entre patrones. Al igual que en la primera aproximación, se utiliza la transformada de Fourier para extraer las características de cada ventana y se normalizan los datos utilizando la media y la desviación típica. El modelo se evaluará utilizando una validación cruzada de 10 folds para garantizar una evaluación más robusta y confiable del modelo.

4.2.2 Resultados:

Durante el entrenamiento de una Red de Neuronas Artificiales (RR.NN.AA.) para clasificar audios de Minecraft como zombie o no zombie, se evaluaron ocho configuraciones distintas. Cada una de estas configuraciones incluía dos neuronas de entrada y una de salida, pero variaba en la cantidad y disposición de las capas ocultas. Estas configuraciones incluyeron una capa oculta con 5, 10, 15 o 20 neuronas, así como varias opciones de dos capas ocultas con diferentes combinaciones de neuronas. La elección de estas configuraciones se debió a que el problema en cuestión tenía pocos patrones y características, por lo que se consideró que una RR.NN.AA. con un número limitado de neuronas en las capas ocultas sería suficiente para obtener un buen rendimiento de clasificación. El objetivo de evaluar estas configuraciones fue encontrar la que obtuviera el mejor resultado en términos de precisión de clasificación. En todos los casos, se estableció el número de folds en 10.

En la tabla 10 se muestra un resumen de los resultados obtenidos para cada configuración. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

Configuración	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
2-5-1	71.12	13.52	62.96	24.31	0.11	1.00	0.80
2-10-1	71.85	13.09	64.19	22.76	0.00	1.00	1.00
2-15-1	71.97	12.45	64.80	21.45	0.00	1.00	1.00
2-20-1	71.12	12.35	64.89	20.51	0.11	1.00	0.80
2-5-15-1	71.35	13.42	63.25	24.60	0.11	1.00	0.80
2-10-15-1	71.59	13.30	64.29	22.62	0.00	1.00	1.00
2-15-15-1	71.38	13.30	63.79	23.98	0.11	1.00	0.80
2-15-20-1	70.96	12.79	64.00	22.77	0.09	1.00	0.86

Table 10: Tabla de resultados RR.NN.AA. 2

La tabla 11 se muestra la matriz de confusión para la configuración 2-15-1 (15 neuronas ocultas en una sola capa), que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	5	0	36	6
Zombie	0	4	6	27

Table 11: Matriz de Confusión de la 3ª iteración (Configuración : 2-15-1)

En el proceso de entrenamiento de SVM hemos utilizado 8 configuraciones distintas, variando los parámetros de kernel, gamma y C. Hemos utilizado el kernel "rbf", ya que es uno de los más utilizados en problemas de clasificación y es adecuado para datos no linealmente separables. Hemos probado con distintos valores de gamma y C, que son hiperparámetros que influyen en la complejidad del modelo y en su capacidad para generalizar. De nuevo, hemos elegido valores de C relativamente altos y gamma bajos o medios para evitar el sobreajuste y mejorar la capacidad de generalización del modelo. A través de esta exploración de hiperparámetros, hemos tratado de encontrar la configuración óptima para nuestro problema de clasificación de sonidos. En todos los casos, hemos establecido el número de folds a 10.

En la tabla 12 se muestra un resumen de los resultados obtenidos para cada configuración. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

kernel	**gamma**	**C**	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
rbf	0.01	100	76.33	13.41	74.06	14.84	0.11	1.00	0.8
rbf	0.01	1000	82.67	8.89	81.94	7.31	0.1	0.66	1.0
rbf	0.01	10000	78.22	10.27	74.77	15.95	0.1	0.66	1.0
rbf	0.001	10000	78.44	6.35	71.52	12.69	0.11	1.0	0.8
rbf	0.0001	10000	72.11	15.02	67.40	16.53	0.11	0.75	1.0
rbf	0.0001	1000	69.89	14.82	64.37	19.59	0.11	1.0	0.8
rbf	0.001	100000	75.11	6.83	71.51	13.02	0.11	0.8	1.0
rbf	0.0001	100000	75.22	14.70	72.07	16.11	0.0	1.0	1.0

Table 12: Tabla de resultados SVM 2

La tabla 13 muestra la matriz de confusión para la configuración con kernel: rbf, gamma: 0.01, C: 1000, que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	7	0	40	0
Zombie	1	2	1	41

Table 13: Matriz de Confusión de la 2ª iteración (kernel: rbf, gamma: 0.01, C: 1000)

En la segunda aproximación del modelo de árboles de decisión, hemos realizado pruebas con seis configuraciones diferentes, ajustando la profundidad máxima del árbol desde 1 hasta 6. Consideramos que este rango de valores es apropiado para resolver nuestro problema, puesto que tenemos un tamaño de datos moderado, con 92 ejemplos y 4 características cada uno. Al explorar diferentes profundidades, nuestro objetivo era encontrar un equilibrio entre la complejidad del modelo y su capacidad para detectar patrones en los datos, evitando tanto el sobreajuste como el subajuste. Además, en todas las pruebas, hemos utilizado un número de folds de 10.

En la tabla 14 se muestra un resumen con los resultados de todas las iteraciones, P se corresponde a la profundidad. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

P	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
1	67.33	13.74	71.99	10.03	0.11	1.00	0.8
2	65.11	16.40	65.39	24.70	0.11	1.00	0.8
3	64.11	7.33	54.71	12.99	0.70	0.66	0.14
4	76.20	9.65	76.09	11.08	0.11	1.00	0.8
5	67.67	11.43	66.16	18.04	0.11	1.00	0.8
6	73.00	15.36	73.16	16.25	0.11	0.80	1.0

Table 14: Tabla de resultados Arboles 2

La tabla 15 muestra las matriz de confusión para profundidad 4, que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	4	1	34	8
Zombie	0	4	3	38

Table 15: Matriz de Confusión de la 4ª Iteración (Profundidad = 4)

En la segunda aproximación del algoritmo K-Nearest Neighbors (KNN), hemos probado seis configuraciones diferentes para el número de vecinos, variando de 1 a 6. Consideramos que estas opciones son apropiadas para nuestro conjunto de datos, ya que cuenta con pocos ejemplos y características. Al utilizar un número bajo de vecinos, estamos limitando la búsqueda y evitando el sobreajuste del modelo, lo que permite que KNN generalice mejor y produzca modelos más precisos para este tipo de datos. En todas las pruebas, hemos utilizado un número de folds de 10.

En la tabla 16 se muestran los resultados de las diferentes iteraciones de k. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

k	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
1	68.78	17.17	61.72	27.75	0.11	1.00	0.80
2	63.22	13.80	47.45	24.47	0.40	0.83	0.25
3	68.67	12.29	64.99	17.60	0.81	0.17	0.21
4	64.33	14.54	56.02	20.38	0.33	1.00	0.50
5	66.44	13.53	62.16	17.68	0.11	1.00	0.80
6	66.44	10.03	60.94	14.06	0.33	1.00	0.50

Table 16: Tabla de resultados KNN 2

La tabla 17 muestra la matriz de confusión de k = 3, que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	3	2	9	33
Zombie	0	4	34	7

Table 17: Matriz de Confusión de la 4ª Iteración (k = 3)

4.2.3 Discusión:

Tras analizar los resultados obtenidos, podemos concluir que la técnica que nos ha proporcionado los mejores resultados ha sido SVM, dado que se alcanzó la mayor precisión media de clasificación y valor F1 medio. En comparación con la primera aproximación, los resultados han mejorando bastante, pasando de un valor medio de precisión del 73.88 a uno del 82.67.

El resto de técnicas también mejoran ligeramente a excepción de la de Árboles de Decisión, por lo que concluimos que ha sido una buena decisión la inclusión de las nuevas características.

Con respecto a la siguiente aproximación, pensamos que lo mejor sería aumentar el tamaño del conjunto de datos. Para ello introduciríamos nuevos sonidos, tanto de zombies como de otros mobs que se clasificarían como no zombies. De esta forma, es probable que el modelo tenga una mayor capacidad para diferenciar entre los diferentes tipos de sonidos y, al tener más información de cada clase de audio.

A continuación se muestra la tabla 18 con el resumen de los mejores resultados de cada modelo:

Técnica	**Average test accuracy**	**Average test F1**	**Configuración**
RR.NN.AA.	71.97	64.80	2-15-1
SVM	82.67	81.94	Kernel = rbf Gamma = 0.01 C = 1000
ARBOLES	76.20	76.09	Profundidad: 4
KNN	68.67	64.99	k = 3

Table 18: Tabla de los mejores resultados de cada técnica de la segunda aproximación

4.3 Aproximación 3 :

4.3.1 Descripción:

En esta tercera aproximación, ampliamos la base de datos para incluir más mobs tanto zombies como no zombies, con el objetivo de mejorar la precisión y generalización de nuestro modelo. Añadimos audios de nuevos tipos de mobs, como el drowned y el zombie villager para la categoría de zombies, y otros como el lobo y el esqueleto para la categoría de no zombies. Esto nos permite obtener un total de 95 fragmentos de audio de zombies y 90 fragmentos de audio de no zombies.

Al igual que en las aproximaciones anteriores, empleamos la FFT (Transformada de Fourier) para extraer las características de cada ventana, y utilizamos la media y la desviación típica para normalizar los datos. En esta aproximación, seguimos utilizando las cuatro características descritas en la aproximación 2: la media y la desviación típica de toda la señal en frecuencia, así como la media y la desviación típica de un fragmento seleccionado de la frecuencia (desde los 1345.8 Hz hasta los 6858.32Hz).

Con este conjunto de datos ampliado, esperamos que nuestro modelo sea capaz de adaptarse mejor a la variabilidad en los audios de distintos mobs y, por lo tanto, mejore su capacidad para distinguir entre zombies y no zombies. Para evaluar la efectividad de nuestro modelo en esta tercera aproximación, continuaremos utilizando una validación cruzada de 10 folds, lo que nos permitirá obtener una evaluación sólida y confiable del rendimiento del modelo.

4.3.2 Resultados:

En primer lugar, durante el entrenamiento de una red de neuronas artificiales (RR.NN.AA.) para clasificar audios de Minecraft como zombie o no zombie, se evaluaron ocho configuraciones distintas para esta tercera aproximación. Cada una de estas configuraciones incluía dos neuronas de entrada y una de salida, pero variaba en la cantidad y disposición de las capas ocultas. Estas configuraciones incluyeron una capa oculta con 5, 10, 15 o 20 neuronas, así como varias opciones de dos capas ocultas con diferentes combinaciones de neuronas. La elección de estas configuraciones se debió a que el problema en cuestión tenía pocos patrones y características, por lo que se consideró que una RR.NN.AA. con un número limitado de neuronas en las capas ocultas sería suficiente para obtener un buen rendimiento de clasificación. El objetivo de evaluar estas configuraciones fue encontrar la que obtuviera el mejor resultado en términos de precisión de clasificación. En todos los casos, se estableció el número de folds en 10.

En la tabla 37 se muestra un resumen de los resultados obtenidos para cada configuración. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

Configuración	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
2-5-1	59.03	7.84	60.05	8.01	0.31	0.72	0.66
2-10-1	64.53	6.77	64.66	8.45	0.47	0.82	0.26
2-15-1	65.50	7.26	65.61	9.29	0.31	0.76	0.62
2-20-1	65.41	6.63	64.89	8.10	0.30	0.81	0.59
2-5-15-1	58.89	6.26	58.26	7.27	0.32	0.58	0.76
2-10-15-1	61.70	7.33	60.98	7.94	0.32	0.88	0.49
2-15-15-1	63.31	6.04	61.58	8.27	0.32	0.58	0.76
2-15-20-1	62.34	5.29	60.28	7.27	0.61	0.083	1.0

Table 19: Tabla de resultados RR.NN.AA. 3

En la tabla 38 se muestra la matriz de confusión para la configuración 2-15-1 (15 neuronas ocultas en 1 capa), que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	5	2	55	33
Zombie	1	10	19	60

Table 20: Matriz de Confusión de la 7ª iteración (Configu ración : 2-15-1)

En el proceso de entrenamiento de SVM hemos utilizado 8 configuraciones distintas, variando los parámetros de kernel, gamma y C. Hemos utilizado el kernel "rbf", ya que es uno de los más utilizados en problemas de clasificación y es adecuado para datos no linealmente separables. Hemos probado con distintos valores de gamma y C, que son hiperparámetros que influyen en la complejidad del modelo y en su capacidad para generalizar. De nuevo, hemos elegido valores de C relativamente altos y gamma bajos o medios para evitar el sobreajuste y mejorar la capacidad de generalización del modelo. A través de esta exploración de hiperparámetros, hemos tratado de encontrar la configuración óptima para nuestro problema de clasificación de sonidos. En todos los casos, hemos establecido el número de folds a 10.

En la tabla 21 se muestra un resumen de los resultados obtenidos para cada configuración. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

kernel	**gamma**	**C**	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
rbf	0.01	100	77.31	6.18	75.90	7.43	0.11	0.91	0.86
rbf	0.01	1000	77.81	6.03	76.22	4.63	0.16	0.83	0.85
rbf	0.01	10000	72.89	6.92	71.99	6.99	0.10	1.00	0.85
rbf	0.001	10000	79.47	6.05	78.90	7.58	0.11	1.0	0.80
rbf	0.0001	10000	76.78	10.40	77.22	11.75	0.05	1.0	0.89
rbf	0.0001	1000	70.29	9.42	71.81	10.68	0.10	0.90	0.89
rbf	0.0001	100000	76.81	10.65	76.48	12.31	0.056	1.00	0.86
rbf	0.001	100000	78.45	8.47	77.17	9.11	0.11	1.00	0.82

Table 21: Tabla de resultados SVM 3

La tabla 22 muestra la matriz de confusión para la configuración con kernel: rbf, gamma: 0.001, C: 10000, que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	8	2	9	76
Zombie	0	9	74	7

Table 22: Matriz de Confusión de la 4ª iteración (kernel: rbf, gamma: 0.001, C: 10000)

En la tercera aproximación del modelo de árboles de decisión, hemos realizado pruebas con seis configuraciones diferentes, ajustando la profundidad máxima del árbol desde 1 hasta 6. Consideramos que este rango de valores es apropiado para resolver nuestro problema, puesto que seguimos teniendo un tamaño de datos moderado, con 185 patrones y 4 características cada uno. Al explorar diferentes profundidades, nuestro objetivo era encontrar un equilibrio entre la complejidad del modelo y su capacidad para detectar patrones en los datos, evitando tanto el sobreajuste como el subajuste. Además, en todas las pruebas, hemos utilizado un número de folds de 10.

En la tabla 23 se muestra un resumen con los resultados de todas las iteraciones, P se corresponde a la profundidad. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

P	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
1	59.59	15.96	68.18	14.21	0.11	1.0	0.6
2	67.16	10.41	70.52	12.07	0.17	0.85	0.8
3	70.32	6.56	67.09	10.90	0.17	0.91	0.71
4	70.29	8.59	69.70	10.73	0.11	1.0	0.71
5	71.34	9.05	69.18	11.04	0.05	1.0	0.86
6	66.96	4.45	65.68	8.11	0.26	0.78	0.7

Table 23: Tabla de resultados Arboles 3

La tabla 24 muestra las matriz de confusión para profundidad 5, que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	6	1	12	76
Zombie	63	16	0	11

Table 24: Matriz de Confusión de la 3ª Iteración (Profundidad = 5)

Para el algoritmo K-Nearest Neighbors (KNN) hemos utilizado seis configuraciones diferentes para el número de vecinos, de 1 a 6. Creemos que estas son opciones adecuadas dado que seguimos teniendo un conjunto de datos con pocos patrones y pocas características. Al utilizar un número bajo de vecinos, estamos limitando el espacio de búsqueda y evitando el sobreajuste del modelo, de esta forma, KNN es capaz de generalizar mejor y producir modelos más precisos para este tipo de datos. En todos los casos, hemos establecido el número de folds a 10.

En la tabla 25 se muestran los resultados de las diferentes iteraciones de k. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

k	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
1	64.85	4.66	61.83	11.71	0.28	1.00	0.58
2	59.88	15.11	44.85	18.06	0.18	0.67	0.92
3	64.23	8.13	62.98	8.23	0.28	0.82	0.57
4	61.52	12.53	51.57	12.84	0.83	0.33	0.08
5	61.02	6.99	58.74	8.48	0.63	0.40	0.33
6	60.49	11.00	55.13	9.82	0.26	0.67	0.78

Table 25: Tabla de resultados KNN 3

La tabla 26 muestra la matriz de confusión de k = 3, que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	4	3	15	73
Zombie	2	9	63	16

Table 26: Matriz de Confusión de la 7ª Iteración (k = 3)

4.3.3 Discusión:

Durante esta tercera fase de nuestro estudio, se ha comprobado que, SVM es la técnica más efectiva de las cuatro utilizadas y KNN la peor. También hemos podido comprobar que la adición de nuevos patrones ha puesto en evidencia las carencias del sistema a la hora de clasificar, ya que es posible que las zonas del espacio de patrones donde están situados esos patrones no estuviesen cubiertas en las aproximaciones anteriores, y ahora sí, y por lo tanto estemos intentando que los modelos aprendan funciones más complejas. Cuando se añaden nuevos patrones al conjunto de validación, puede ocurrir que los modelos de aprendizaje automático entrenados previamente no sean capaces de clasificarlos correctamente. Esto puede deberse a diferentes razones, como la complejidad del problema de clasificación, la falta de representatividad de los patrones utilizados en el entrenamiento, o la falta de características relevantes para la clasificación. Esto significa que los modelos entrenados anteriormente no han sido capaces de capturar las características relevantes de los patrones nuevos, lo que ha llevado a una mala clasificación. Por otro lado, la inclusión de los audios del lobo como no zombie también ha podido dificultar la clasificación, puesto que el sonido es similar al de los zombies.

Tras analizar los resultados obtenidos, creemos que se requiere la inclusión de características más determinantes en el modelo, que permitan una mejor discriminación entre las clases de audio, para lograr una correcta clasificación entre zombies y no zombies. Concretamente, al igual que en la aproximación 2, incluiremos la media y desviación típica de nuevos trozos de frecuencia para intentar captar mejor las diferencias entre clases de audio.

A continuación se muestra la tabla 45 con el resumen de los mejores resultados de cada modelo:

Técnica	**Average test accuracy**	**Average test F1**	**Configuración**
RR.NN.AA.	65.50	65.61	2-15-1
SVM	79.47	78.90	Kernel = rbf Gamma = 0.001 C = 10000
ARBOLES	71.34	69.18	Profundidad: 5
KNN	64.23	62.98	k = 3

Table 27: Tabla de los mejores resultados de cada técnica de la tercera aproximación

4.4 Aproximación 4:

4.4.1 Descripción:

En la tercera aproximación, observamos que al aumentar el número de patrones en la base de datos, se detectó que había zonas del espacio que no estaban cubiertas por nuestras características. Como consecuencia, los resultados empeoraron en todas las técnicas. Para abordar este problema en la cuarta aproximación, decidimos incluir cuatro características adicionales.
Estas nuevas características corresponden a la media y desviación típica de dos nuevos segmentos de frecuencia: 5512.5Hz a 11025Hz y 9679.17Hz a 15191.67Hz. Estos segmentos se eligieron después de examinar las gráficas y determinar que era donde había una mayor variación de la frecuencia.
Al igual que en las aproximaciones anteriores, utilizamos la transformada de Fourier para extraer las características de cada ventana y normalizamos los datos utilizando la media y la desviación típica. Para evaluar el modelo, utilizamos una validación cruzada de 10 folds, lo que garantiza una evaluación más robusta y confiable del mismo.

4.4.2 Resultados:

Durante el entrenamiento de una Red de Neuronas Artificiales (RR.NN.AA.) para clasificar audios de Minecraft como zombie o no zombie, se evaluaron ocho configuraciones distintas. Cada una de estas configuraciones incluía dos neuronas de entrada y una de salida, pero variaba en la cantidad y disposición de las capas ocultas. Estas configuraciones incluyeron una capa oculta con 5, 10, 15 o 20 neuronas, así como varias opciones de dos capas ocultas con diferentes combinaciones de neuronas. La elección de estas configuraciones se debió a que el problema en cuestión sigue teniendo pocos patrones y características, por lo que se consideró que una RR.NN.AA. Con un número limitado de neuronas en las capas ocultas sería suficiente para obtener un buen rendimiento de clasificación. El objetivo de evaluar estas configuraciones fue encontrar la que obtuviera el mejor resultado en términos de precisión de clasificación. En todos los casos, se estableció el número de folds en 10.

En la tabla 28 se muestra un resumen de los resultados obtenidos para cada configuración. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

Configuración	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
2-5-1	75.21	8.56	74.35	10.5	0.16	0.89	0.8
2-10-1	76.45	8.16	75.17	9.89	0.05	0.89	1.0
2-15-1	76.07	8.48	74.57	10.27	0.11	0.89	0.90
2-20-1	75.25	8.09	73.71	10.38	0.053	0.89	1.0
2-5-15-1	73.46	8.40	71.91	10.6	0.11	0.89	0.9
2-10-15-1	75	7.61	73.05	9.95	0.16	0.89	0.8
2-15-15-1	73.95	7.8	71.76	10.09	0.11	0.78	1.0
2-15-20-1	73.28	7.80	71.29	9.77	0.05	0.89	1.0

Table 28: Tabla de resultados RR.NN.AA. 4

La tabla 29 se muestra la mejor matriz de confusión para la configuración 2-10-1 (10 neuronas ocultas en una sola capa), que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	10	0	69	16
Zombie	1	8	19	62

Table 29: Matriz de Confusión de la 2ª iteración (Configuración : 2-10-1)

En el proceso de entrenamiento de SVM hemos utilizado 8 configuraciones distintas, variando los parámetros de kernel, gamma y C. Hemos utilizado el kernel "rbf", ya que es uno de los más utilizados en problemas de clasificación y es adecuado para datos no linealmente separables. Hemos probado con distintos valores de gamma y C, que son hiperparámetros que influyen en la complejidad del modelo y en su capacidad para generalizar. De nuevo, hemos elegido valores de C relativamente altos y gamma bajos o medios para evitar el sobreajuste y mejorar la capacidad de generalización del modelo. A través de esta exploración de hiperparámetros, hemos tratado de encontrar la configuración óptima para nuestro problema de clasificación de sonidos. En todos los casos, hemos establecido el número de folds a 10.

En la tabla 30 se muestra un resumen de los resultados obtenidos para cada configuración. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

kernel	**gamma**	**C**	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
rbf	0.01	100	80.0	7.51	78.50	10.01	0.11	0.89	0.9
rbf	0.01	1000	79.50	10.26	77.35	13.63	0.11	0.88	0.9
rbf	0.01	10000	81.17	12.50	79.69	14.77	0.0	1.00	1.00
rbf	0.001	10000	80.09	8.26	78.72	10.29	0.06	0.83	1.0
rbf	0.0001	10000	78.42	6.99	77.74	8.27	0.10	0.89	0.9
rbf	0.0001	1000	69.18	11.49	71.25	9.59	0.21	0.89	0.7
rbf	0.0001	100000	83.25	3.94	83.48	4.54	0.11	1.00	0.8
rbf	0.001	100000	81.08	9.94	78.27	13.94	0.05	1.00	0.9

Table 30: Tabla de resultados SVM 4

La tabla 31 muestra la matriz de confusión para la configuración con kernel: rbf, gamma: 0.0001, C: 100000, que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	8	2	71	14
Zombie	0	9	9	72

Table 31: Matriz de Confusión de la 4ª iteración (kernel: rbf, gamma: 0.0001, C: 100000)

En la cuarta aproximación del modelo de árboles de decisión, hemos realizado pruebas con seis configuraciones diferentes, ajustando la profundidad máxima del árbol desde 8 hasta 13, puesto que hemos comprobado que al haber añadido nuevas características, se obtenían mejores resultados con valores de profundidad más altos. Al explorar diferentes profundidades, nuestro objetivo era encontrar un equilibrio entre la complejidad del modelo y su capacidad para detectar patrones en los datos, evitando tanto el sobreajuste como el subajuste. Además, en todas las pruebas, hemos utilizado un número de folds de 10.

En la tabla 41 se muestra un resumen con los resultados de todas las iteraciones, P se corresponde a la profundidad. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

P	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
8	73.57	7.11	71.71	10.14	0.84	0.33	0.0
9	72.95	9.06	71.32	12.58	0.06	0.89	1.0
10	72.95	9.06	71.32	12.58	0.06	0.89	1.0
11	72.95	9.06	71.32	12.58	0.06	0.89	1.0
12	72.95	9.06	71.32	12.58	0.06	0.89	1.0
13	72.95	9.06	71.32	12.58	0.06	0.89	1.0

Table 32: Tabla de resultados Arboles 4

La tabla 42 muestra la mejor matriz de confusión para profundidad 8, que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	7	5	83	2
Zombie	2	5	5	76

Table 33: Matriz de Confusión de la 2ª Iteración (Profundidad = 8)

Para el algoritmo K-Nearest Neighbors (KNN) hemos utilizado seis configuraciones diferentes para el número de vecinos, de 1 a 6. Creemos que estas son opciones adecuadas dado que seguimos teniendo un conjunto de datos con pocos patrones y pocas características y probando con valores más altos no hemos obtenido mejorar los resultados. Al utilizar un número bajo de vecinos, estamos limitando el espacio de búsqueda y evitando el sobreajuste del modelo, de esta forma, KNN es capaz de generalizar mejor y producir modelos más precisos para este tipo de datos. En todos los casos, hemos establecido el número de folds a 10.

En la tabla 34 se muestran los resultados de las diferentes iteraciones de k. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

k	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
1	62.16	5.65	58.40	13.42	0.28	0.82	0.57
2	59.44	7.76	44.64	15.21	0.39	0.50	0.67
3	67.05	5.69	64.41	7.84	0.26	0.67	0.77
4	62.72	9.03	53.47	13.22	0.16	0.67	0.92
5	60.61	7.44	58.58	10.67	0.26	0.67	0.77
6	60.06	8.34	53.73	12.96	0.21	0.67	0.85

Table 34: Tabla de resultados KNN 4

La tabla 35 muestra la matriz de confusión de k = 3, que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	4	3	15	73
Zombie	2	9	63	16

Table 35: Matriz de Confusión de la 3º Iteración (k = 3)

4.4.3 Discusión:

A continuación se muestra la tabla 36 con el resumen de los mejores resultados de cada modelo:

Técnica	**Average test accuracy**	**Average test F1**	**Configuración**
RR.NN.AA.	76.45	75.17	2-10-1
SVM	83.25	83.48	Kernel = rbf Gamma = 0.0001 C = 100000
ARBOLES	73.57	71.71	Profundidad: 8
KNN	67.05	64.41	k = 3

Table 36: Tabla de los mejores resultados de cada técnica de la cuarta aproximación

Observando los resultados de la cuarta aproximación, podemos concluir que la técnica más efectiva ha sido SVM, con una precisión media de clasificación y valor F1 medio más altos. Por otro lado, la técnica KNN fue la peor, con la precisión media más baja y el valor F1 medio más bajo. Además, podemos concluir que las nuevas características han sido útiles, ya que han mejorado los resultados en todas las técnicas. En particular, RR.NN.AA. tuvo un aumento significativo en el rendimiento, lo que sugiere que estas nuevas características proporcionan información relevante para la clasificación de sonidos de mobs de Minecraft.

4.5 Aproximación 5 :

4.5.1 Descripción:

En esta aproximación extra, ampliamos la base de datos para incluir más mobs no zombies, (dado que ya hemos incluido a todos los zombies) con el objetivo de mejorar la precisión y generalización de nuestro modelo. Concretamente añadimos al pollo y el silverfish. Esto nos permite añadir 10 fragmentos de audio del pollo y 18 fragmentos de audio del silverfish, que se suman a los 90 fragmentos de audio de no zombies que ya teníamos previamente.

Al igual que en las aproximaciones anteriores, empleamos la FFT (Transformada de Fourier) para extraer las características de cada ventana, y utilizamos la media y la desviación típica para normalizar los datos. En esta aproximación, seguimos utilizando las cuatro características descritas en la aproximación 4: la media y la desviación típica de toda la señal en frecuencia, así como la media y desviación típica de tres segmentos de frecuencia: 1345.8 Hz a 6858.32Hz, 5512.5Hz a 11025Hz y 9679.17Hz a 15191.67Hz

Con este conjunto de datos ampliado, esperamos que nuestro modelo sea capaz de adaptarse mejor a la variabilidad en los audios de distintos mobs y, por lo tanto, mejore su capacidad para distinguir entre zombies y no zombies. Para evaluar la efectividad de nuestro modelo en esta tercera aproximación, continuaremos utilizando una validación cruzada de 10 folds, lo que nos permitirá obtener una evaluación sólida y confiable del rendimiento del modelo.

4.5.2 Resultados:

En primer lugar, durante el entrenamiento de una red de neuronas artificiales (RR.NN.AA.) para clasificar audios de Minecraft como zombie o no zombie, se evaluaron ocho configuraciones distintas para esta tercera aproximación. Cada una de estas configuraciones incluía dos neuronas de entrada y una de salida, pero variaba en la cantidad y disposición de las capas ocultas. Estas configuraciones incluyeron una capa oculta con 5, 10, 15 o 20 neuronas, así como varias opciones de dos capas ocultas con diferentes combinaciones de neuronas. La elección de estas configuraciones se debió a que el problema en cuestión tenía pocos patrones y características, por lo que se consideró que una RR.NN.AA. con un número limitado de neuronas en las capas ocultas sería suficiente para obtener un buen rendimiento de clasificación. El objetivo de evaluar estas configuraciones fue encontrar la que obtuviera el mejor resultado en términos de precisión de clasificación. En todos los casos, se estableció el número de folds en 10.

En la tabla 37 se muestra un resumen de los resultados obtenidos para cada configuración. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

Configuración	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
2-5-1	78.05	7.21	72.31	12.43	0.09	1.0	0.87
2-10-1	77.88	8.27	71.73	12.64	0.09	1.0	0.87
2-15-1	77.21	8.57	70.26	14.39	0.09	1.0	0.87
2-20-1	76.64	9.65	68.95	15.18	0.04	1.0	0.93
2-5-15-1	74.54	8.81	66.54	13.06	0.09	1.0	0.87
2-10-15-1	74.75	9.85	66.36	15.54	0.09	1.0	0.87
2-15-15-1	74.84	10.45	66.41	16.95	0.09	1.0	0.87
2-15-20-1	74.13	10.51	65.12	17.16	0.14	0.83	0.87

Table 37: Tabla de resultados RR.NN.AA. 3

En la tabla 38 se muestra la matriz de confusión para la configuración 2-5-1 (5 neuronas ocultas en 1 capa), que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	14	2	94	13
Zombie	0	6	17	67

Table 38: Matriz de Confusión de la 3ª iteración (Configuración : 2-5-1)

Para esta aproximación, en el proceso de entrenamiento de SVM hemos utilizado 8 configuraciones distintas, variando los parámetros de kernel, gamma y C. Hemos utilizado el kernel "rbf", ya que es uno de los más utilizados en problemas de clasificación y es adecuado para datos no linealmente separables. Hemos probado con distintos valores de gamma y C, que son hiperparámetros que influyen en la complejidad del modelo y en su capacidad para generalizar. De nuevo, hemos elegido valores de C relativamente altos y gamma bajos o medios para evitar el sobreajuste y mejorar la capacidad de generalización del modelo. A través de esta exploración de hiperparámetros, hemos tratado de encontrar la configuración óptima para nuestro problema de clasificación de sonidos. En todos los casos, hemos establecido el número de folds a 10.

En la tabla 39 se muestra un resumen de los resultados obtenidos para cada configuración. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

kernel	**gamma**	**C**	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
rbf	0.01	100	80.00	7.5	78.5	10.0	0.10	0.89	0.90
rbf	0.01	1000	79.5	10.26	77.35	13.63	0.10	0.89	0.90
rbf	0.01	10000	81.16	12.49	79.69	14.77	0.00	1.00	1.00
rbf	0.001	10000	80.08	8.26	78.72	10.29	0.05	0.83	1.00
rbf	0.0001	10000	78.42	6.99	77.74	8.27	0.10	0.89	0.90
rbf	0.0001	1000	67.07	14.40	71.78	12.13	0.10	1.00	0.80
rbf	0.0001	100000	66.08	14.68	70.91	13.67	0.11	1.00	0.60
rbf	0.001	100000	68.13	8.82	68.53	12.26	0.16	0.89	0.80

Table 39: Tabla de resultados SVM 6

La tabla 40 muestra la matriz de confusión para la configuración con kernel: rbf, gamma: 0.01, C: 10000, que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	7	0	83	5
Zombie	0	11	3	78

Table 40: Matriz de Confusión de la 3ª iteración (kernel: rbf, gamma: 0.01, C: 10000)

En la quinta aproximación del modelo de árboles de decisión, al igual que en la cuarta, hemos realizado pruebas con seis configuraciones diferentes, ajustando la profundidad máxima del árbol desde 8 hasta 13. Al explorar diferentes profundidades, nuestro objetivo era encontrar un equilibrio entre la complejidad del modelo y su capacidad para detectar patrones en los datos, evitando tanto el sobreajuste como el subajuste. Además, en todas las pruebas, hemos utilizado un número de folds de 10.

En la tabla 41 se muestra un resumen con los resultados de todas las iteraciones, P se corresponde a la profundidad. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

P	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
8	70.97	6.46	62.64	15.61	0.8	0.22	0.16
9	70.49	6.56	63.99	15.71	0.8	0.11	0.25
10	70.93	5.52	63.86	12.97	0.8	0.11	0.25
11	70.95	5.79	63.83	13.74	0.8	0.11	0.25
12	71.90	5.79	65.32	13.94	0.8	0.11	0.25
13	71.90	5.79	65.32	13.94	0.8	0.11	0.25

Table 41: Tabla de resultados Arboles 4

La tabla 42 muestra la mejor matriz de confusión:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	3	9	111	0
Zombie	8	1	0	81

Table 42: Matriz de Confusión de la 6ª Iteración

Para el algoritmo K-Nearest Neighbors (KNN) hemos utilizado seis configuraciones diferentes para el número de vecinos, de 1 a 6. Creemos que estas son opciones adecuadas dado que seguimos teniendo un conjunto de datos con pocos patrones y pocas características. Al utilizar un número bajo de vecinos, estamos limitando el espacio de búsqueda y evitando el sobreajuste del modelo, de esta forma, KNN es capaz de generalizar mejor y producir modelos más precisos para este tipo de datos. En todos los casos, hemos establecido el número de folds a 10.

En la tabla 43 se muestran los resultados de las diferentes iteraciones de k. Se incluyen los valores de error, sensibilidad y especificidad de test de la mejor iteración.

k	**Average test accuracy**	**Accuracy standard deviation**	**Average test F1**	**Standard deviation F1**	**Error**	**Sensibilidad**	**Especificidad**
1	74.16	6.11	60.09	12.26	0.0	1.0	1.0
2	74.16	6.11	60.09	12.26	0.1	0.77	1.0
3	72.25	9.08	66.13	12.19	0.14	0.9	0.83
4	71.75	6.21	59.58	7.62	0.27	0.5	0.92
5	72.66	7.64	66.64	6.75	0.23	0.8	0.75
6	72.21	8.06	62.06	8.56	0.38	0.56	0.67

Table 43: Tabla de resultados KNN 6

La tabla 44 muestra la matriz de confusión de k = 1, que nos ha devuelto los mejores resultados:

	Test		**Training**	
	Clasificado como no zombie	Clasificado como zombie	Clasificado como no zombie	Clasificado como zombie
No zombie	12	2	109	0
Zombie	3	5	0	82

Table 44: Matriz de Confusión de la 6ª Iteración (k = 1)

4.5.3 Discusión:

Durante esta quinta fase de nuestro estudio, se ha comprobado que, SVM vuelve a ser la técnica más efectiva de las cuatro utilizadas pero en este caso la de árboles de decisión es la peor. Al aumentar el número de patrones en la base de datos hemos observado en general una mejora en los resultados, esto sugiere que el modelo de aprendizaje automático está aprendiendo más de los datos y mejorando su capacidad de generalización.

Agregar más patrones a la base de datos aumenta la cantidad de información que el modelo tiene para aprender y le permite detectar patrones y características más precisas que antes podrían haber estado ocultas. Esto a su vez puede mejorar su capacidad para clasificar con mayor precisión las nuevas instancias de datos que nunca antes ha visto.

A continuación se muestra la tabla 45 con el resumen de los mejores resultados de cada técnica:

Técnica	**Average test accuracy**	**Average test F1**	**Configuración**
RR.NN.AA.	78.05	72.31	2-5-1
SVM	81.16	79.69	Kernel = rbf Gamma = 0.01 C = 10000
ARBOLES	71.90	65.32	Profundidad: 12
KNN	74.16	60.09	k = 1

Table 45: Tabla de los mejores resultados de cada técnica de la quinta aproximación

5 Deep Learning

5.1 Descripción

En esta aproximación nos centraremos en Deep Learning. A diferencia de las aproximaciones anteriores, en esta solo usaremos una técnica, redes neuronales convolucionales (CNN). En Deep Learning se suele trabajar con bases de datos enormes pero en nuestro caso será pequeña, concretamente la misma que hemos utilizado en la cuarta aproximación. Al igual que en las anteriores aproximaciones, fragmentamos los audios de la base de datos. Para realizar esta fragmentación, como la frecuencia de los audios es de 44100 muestras por segundo, empleamos un tamaño de ventana de 2^{15} muestras (32768 muestras, aproximadamente medio segundo) y además establecemos un porcentaje de solapamiento de 0.5.
Para poder trabajar con CNN, obtenemos dos arrays, un array con los arrays de las señales de frecuencia de los distintos fragmentos y otro con las salidas esperadas. Después separamos los datos para entrenamiento y test. Para procesar los audios con Deep Learning, los pasamos a una matriz en formato HWCN, es decir, Height x Width x Channels x N (en nuestro caso Width=1 y Channels=1). A continuación, normalizamos las señales. En nuestro trabajo, optamos por
realizar la normalización de señales de frecuencia utilizando el valor máximo y mínimo porque es una técnica sencilla y efectiva para asegurar que todas las señales tengan la misma escala y no se vean afectadas por valores extremos. Además, nos permite comparar y analizar los datos de manera más eficiente y precisa.
Por último para realizar el entrenamiento, cuando se tienen muchos patrones de entrenamiento generalmente no se entrena pasando todos los patrones y modificando el error, en su lugar, el conjunto de entrenamiento se divide en subconjuntos (batches) y se van aplicando uno a uno. Como en nuestro caso tenemos pocos patrones, utilizaremos un único batch.
Utilizaremos 8 arquitecturas distintas que se especificarán en el siguiente apartado.

5.2 Resultados

En la tabla 46 se muestran los parámetros usados en cada capa para la primera arquitectura:

Configuración	Resultados
Conv((3, 1), 1 =>16, MaxPool((2, 1)) Conv((3, 1), 16 =>32, MaxPool((2, 1)) Conv((3, 1), 32 =>32, MaxPool((2, 1))	Da de máxima precisión de test 60.24 con una precisión de entrenamiento de 52.63

Table 46: Primera arquitectura.

En la tabla 47 se muestran los parámetros usados en cada capa para la segunda arquitectura:

Configuración	Resultados
Conv((5, 1), 1 =>20, MaxPool((2, 1)) Conv((3, 1), 20 =>40, MaxPool((2, 1)) Conv((3, 1), 40 =>40, MaxPool((2, 1))	Da de máxima precisión de test 60.24 con una precisión de entrenamiento de 52.63

Table 47: Segunda arquitectura.

En la tabla 48 se muestran los parámetros usados en cada capa para la tercera arquitectura:

Configuración	Resultados
Conv((3, 1), 1 =>16, MaxPool((2, 1)) Conv((3, 1), 16 =>32, MaxPool((2, 1)) Conv((3, 1), 32 =>32, MaxPool((2, 1))	Da de máxima precisión de test 57.22 con una precisión de entrenamiento de 57.89

Table 48: Tercera arquitectura.

En la tabla 49 se muestran los parámetros usados en cada capa para la cuarta arquitectura:

Configuración	Resultados
Conv((5, 1), 1 =>16, MaxPool((3, 1))	Da de máxima precisión de test 60.24
Conv((5, 1), 16 =>32, MaxPool((3, 1))	con una precisión de entrenamiento de 52.63
Conv((3, 1), 32 =>32, MaxPool((2, 1))	

Table 49: Cuarta arquitectura.

En la tabla 50 se muestran los parámetros usados en cada capa para la quinta arquitectura:

Configuración	Resultados
Conv((7, 1), 1 =>16, MaxPool((2, 1))	Da de máxima precisión de test 60.24
Conv((5, 1), 16 =>32, MaxPool((2, 1))	con una precisión de entrenamiento de 52.63
Conv((3, 1), 32 =>64, MaxPool((2, 1))	

Table 50: Quinta arquitectura.

En la tabla 51 se muestran los parámetros usados en cada capa para la sexta arquitectura:

Configuración	Resultados
Conv((3, 1), 1 =>24, MaxPool((2, 1))	Da de máxima precisión de test 60.24
Conv((3, 1), 24 =>48, MaxPool((2, 1))	con una precisión de entrenamiento de 52.63
Conv((3, 1), 48 =>48, MaxPool((2, 1))	

Table 51: Sexta arquitectura.

En la tabla 52 se muestran los parámetros usados en cada capa para la primera arquitectura:

Configuración	Resultados
Conv((4, 1), 1 =>20 , MaxPool((3, 1)) Conv((4, 1), 20 =>40, MaxPool((3, 1)) Conv((4, 1), 40 =>40, MaxPool((2, 1))	Da de máxima precisión de test 60.24 con una precisión de entrenamiento de 52.63

Table 52: Séptima arquitectura

Se muestra la tabla 53 con los parámetros usados en cada capa para la octava arquitectura:

Configuración	Resultados
Conv((3, 1), 1 =>12, MaxPool((2, 1)) Conv((3, 1), 12 =>24, MaxPool((2, 1)) Conv((3, 1), 24 =>24,MaxPool((2, 1))	Da de máxima precisión de test 60.24 con una precisión de entrenamiento de 52.63

Table 53: Octava arquitectura

5.3 Discusión

Al analizar los resultados obtenidos, se observa que todas las arquitecturas exhiben un rendimiento similar, alcanzando una precisión del 60.24% en el conjunto de entrenamiento y del 52.63% en el conjunto de pruebas. Sin embargo,la arquitectura 3, por ejemplo, si que arroja resultados ligeramente diferentes. A pesar de esta variabilidad, todas las arquitecturas convergen finalmente hacia un mismo resultado, logrando una precisión de entrenamiento del 39.76%. Estos hallazgos sugieren que, aunque las arquitecturas inicialmente parecen divergir, su desempeño converge a medida que el entrenamiento progresa.

6 Conclusiones

Tras analizar los resultados obtenidos en este trabajo, podemos concluir que en general han sido buenos. Hemos logrado desarrollar un sistema de detección de audios mediante aprendizaje automático que es capaz de diferenciar mobs de Minecraft por su sonido y detectar cuáles son zombies y cuáles no. En cuanto a los modelos utilizados, hemos comprobado que el mejor de ellos ha sido SVM con un kernel "rbf", gamma 0.01 y valor de C=10000, obteniendo una tasa de acierto superior al 80%.

Si bien es cierto que el tamaño reducido de la base de datos y la duración de los audios han sido dificultades a la hora de desarrollar el sistema, hemos logrado superar estos obstáculos mediante el uso de técnicas de preprocesamiento y extracción de características de los audios.

En cuanto a la viabilidad de utilizar este tipo de sistemas para resolver el problema, consideramos que es una opción muy válida y efectiva, ya que los resultados obtenidos son muy buenos. Sin embargo, también es importante considerar que en otros entornos o problemas puede que sea necesario aplicar nuevas técnicas de aprendizaje automático o combinar diferentes modelos para obtener los mejores resultados.

En resumen, no hemos logrado cumplir los objetivos iniciales del trabajo, ya que pretendíamos hacer un sistema que diferenciara los mobs de minecraft por su sonido y nos quedamos en uno que diferenciase los zombies de los que no son zombies. Esto se debe a que el videojuego en cuestión cuenta con muy pocos audios para cada mob (y a su vez el número de mobs es bastante elevado), lo que hace difícil entrenar a un sistema que sea capaz de distinguirlos a todos (además también nos ha perjudicado que algunos sonidos de mobs distintos sean bastante similares). A pesar de esto, como hemos mencionado anteriormente, con la simplificación del problema hemos logrado conseguir un sistema que clasifica con bastante precisión entre zombie y no zombie.

7 Trabajo futuro

Actualmente, el sistema es capaz de detectar zombies y diferenciarlos de otros mobs, pero lo ideal sería ampliar su capacidad para identificar y distinguir otros tipos de mobs por su sonido.

Para lograr este objetivo, una posible estrategia sería implementar técnicas más avanzadas de aprendizaje automático, como redes neuronales profundas (DNN), que pueden ser más efectivas para reconocer patrones y características sutiles en los datos de audio. Asimismo, también es necesario explorar la posibilidad de utilizar técnicas de procesamiento de señales digitales para extraer características específicas de cada tipo de mob que sean más distintivas y fáciles de detectar.

Por otro lado, nuestro sistema podría expandirse a otros videojuegos, de forma que pudiese distinguir también entidades que no perteneciesen a Minecraft. Para esto habría que ampliar la base de datos con los audios de los nuevos juegos soportados.

Además, sería beneficioso recolectar más datos de audio de Minecraft para entrenar y mejorar el modelo, como por ejemplo audios con ruido de fondo, ya que una mayor cantidad de datos podría ayudar a mejorar la capacidad del modelo para detectar y distinguir diferentes tipos de mobs por su sonido. En resumen, existen diversas áreas de trabajo que se pueden abordar para mejorar el sistema de detección de audio, y la identificación precisa de diferentes tipos de mobs por su sonido podría ser una mejora valiosa para la utilidad del sistema en general.

8 Bibliografía

[1] Minecraft Sound Pack — 1.17. Mine-imator forums. Recuperado el 29 de marzo, 2023, de https://www.mineimatorforums.com/index.php?/topic/46037-minecraft-sound-pack-%E2%80%94-117/

[2] J.M. Fonseca-Solís.(2015, 16 de febrero). *Reconocimiento automático de la altura musical en la interfaz de un juego de computadora*. Revista Ingeniería. Recuperado el 29 de marzo, 2023, de https://www.researchgate.net/publication/282476167_RECONOCIMIENTO_AUTOMATICO_DE_LA_ALTURA_MUSICAL_EN_LA_INTERFAZ_DE_UN_JUEGO_DE_COMPUTADORA

[3] F.J. Bravo Sanchez, Md Rahat Hossain, N.B. English S.T. Moore. (2021, 3 de octubre). *Bioacoustic classification of avian calls from raw sound recordings using convolutional neural networks*, Scientific Reports. Disponible en: https://www.nature.com/articles/s41598-021-95076-6

[4] Nanni, Loris and Brahnam, Sheryl and Lumini, Alessandra and Maguolo, Gianluca. (2020). *Animal Sound Classification Using Dissimilarity Spaces*. Applied Sciences, 10(23), 8578. Disponible en: https://www.mdpi.com/2076-3417/10/23/8578

[5] Eduardo.C. Nunes. (2021, 15 de febrero). *Anomalous Sound Detection with Machine Learning: A Systematic Review*. Disponible en: https://www.researchgate.net/publication/349364033_Anomalous_Sound_Detection_with_Machine_Learning_A_Systematic_Review

[6] Ágnes Incze, Henrietta-Bernadett Jancsó, Zoltán Szilágyi, Attila Farkas, Csaba Sulyok. (2018, 1 de septiembre). *Bird Sound Recognition Using a Convolutional Neural Network*. Disponible en: https://www.researchgate.net/publication/328836649_Bird_Sound_Recognition_Using_a_Convolutional_Neural_Network.

[7] Rodríguez Ramírez, P. (2020). *Clasificación automática de sonidos utilizando aprendizaje máquina*. Universidad de Sevilla, Sevilla. [Trabajo Final de Grado, Universidad de Sevilla]. Disponible en: https://idus.us.es/handle/11441/101412

[8] Luque, J., Larios, D. F., Personal, E., Barbancho, J., and León, C. (2016, 18 de mayo). *Evaluation of mpeg-7- based audio descriptors for animal voice recognition over wireless acoustic sensor networks*. Sensors, 16(5):717. Disponible en: https://idus.us.es/bitstream/handle/11441/48628/Sensors_Luque_2016_Evaluation.pdf?sequence=1

[9] Isha Agarwal, Parul Yadav, Neha Gupta, Sarita Yadav. (2021). *Urban Sound Classification Using Machine Learning and Neural Networks*. Disponible en: https://link.springer.com/chapter/10.1007/978-981-33-4501-0_31