

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“DETECCIÓN DE MENTIRAS EN VIDEOS DE PRESENTACIÓN MEDIANTE MACHINE LEARNING”

BASTIÁN SUZUKI SUZUKI CHÁVEZ

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Liubov Dombrovskaia
Profesor Correferente: Roberto León

Enero - 2025

DEDICATORIA

*Por aquellos momentos
Que se congelan en el tiempo
Que duran solo segundos
Pero en la mente son eternos*

Dedicado a todas aquellas personas comprometidas con generar un impacto positivo en el mundo y contribuir a la construcción de una sociedad mejor.

AGRADECIMIENTOS

Agradezco de todo corazón a mi querida familia, quienes han sido un pilar fundamental en mi vida y en la realización de este proyecto. A mis amados padres René y Ximena, les agradezco infinitamente por su constante cuidado y apoyo incondicional, especialmente en los momentos más difíciles de mi vida personal y profesional. Gracias por enseñarme la importancia de buscar siempre la mejor versión de mí mismo y por recordarme que la salud es uno de los aspectos más importantes de la vida. Son los mejores padres que podría haber tenido, siempre alentándome a seguir adelante y a superar cualquier desafío que se presente.

A mi hermano menor Ignacio, quien es mucho más que un hermano, un mejor amigo, un coach y mucho más. Agradezco su constante aliento y motivación durante todo este proyecto, especialmente en los días más difíciles. Siempre dispuesto a desafiarme a aprender cosas nuevas y a crecer tanto personal como profesionalmente.

A todos los profesores que tuve la oportunidad de conocer durante mi trayectoria universitaria, en especial a los profesores del departamento de informática. También un especial agradecimiento al profesor José Luis Martí, jefe de carrera y profesor de muchos de los ramos que cursé, siempre dispuesto a escuchar y ayudar en los problemas académicos. Asimismo, agradezco a mi profesora guía Liuba por su apoyo durante el desarrollo de este trabajo, por su feedback, sus consejos y por brindarme la oportunidad de aplicar mis conocimientos y pasión en este proyecto.

A mis compañeros de universidad, a los ingenieros civiles eléctricos de la generación 2015, a los compañeros de informática a quienes conocí cuando me cambié de carrera y a cada persona con la que compartí durante mi tiempo en la universidad. También un especial agradecimiento a Franco, uno de mis mejores amigos con quien tuve la oportunidad de compartir y asistir al mismo colegio, la misma universidad y ahora también trabajar en la misma empresa. Estoy agradecido por nuestra amistad y por todos los momentos en que nos hemos apoyado y compartido.

A mis compañeros de trabajo del banco BCI, quienes me brindaron la oportunidad de trabajar mientras aún era estudiante. Agradezco su amabilidad y comprensión al permitirme administrar mi tiempo para cumplir con mis obligaciones académicas.

Y no puedo olvidar mencionar a mi perrita Suki, por sus peticiones de juego y de paseo cuando me encontraba trabajando en este proyecto. Gracias por brindarme momentos de distracción y alegría.

RESUMEN

Resumen— Este trabajo de investigación se centró en abordar el desafío de la detección de mentiras a través del análisis de audios. Se propuso utilizar modelos de Deep Learning como enfoque principal para analizar y determinar la veracidad de los audios. El objetivo fue aplicar e integrar una metodología que permitiera identificar de manera efectiva las mentiras basándose en las características de las señales de audio.

Para lograr este objetivo, se siguieron diez etapas fundamentales de la metodología de la ciencia de datos. Se obtuvo un dataset de videos clasificados previamente en mentiras y verdades, el cual fue utilizado para entrenar y evaluar diferentes modelos de Deep Learning. Se exploraron y compararon modelos como MLP, CNN y RNN, utilizando representaciones de los audios en forma de STFT y MFCC. Tras la evaluación exhaustiva, se seleccionó el modelo RNN con datos MFCC debido a su precisión de 50 % y función de perdida de 0,93 en la detección de mentiras en audios.

Los resultados obtenidos demuestran la eficacia de los modelos de Deep Learning en el análisis de audios y su relevancia en el campo de la detección de mentiras.

Palabras Clave— Transformada de Fourier, Coeficientes Cepstrales en las Frecuencias de Mel, Redes neuronales, Detector de mentiras

ABSTRACT

Abstract— This research work focused on addressing the challenge of lie detection through audio analysis. The proposal was to utilize Deep Learning models as the primary approach for analyzing and determining the veracity of audios. The objective was to apply and integrate a methodology that effectively identifies lies based on the characteristics of audio signals.

To achieve this goal, ten fundamental stages of the data science methodology were followed. A dataset of videos classified as lies and truths was obtained, which was used to train and evaluate different Deep Learning models. Models such as MLP, CNN, and RNN were explored and compared using audio representations in the form of STFT and MFCC. After a thorough evaluation, the RNN model with MFCC data was selected due to its accuracy of 50 % and a loss function of 0,93 in detecting lies in audios.

The obtained results demonstrate the effectiveness of Deep Learning models in audio analysis and their relevance in the field of lie detection.

Keywords— Fourier transform, cepstral coefficients at Mel frequencies, neural networks, lie detector

GLOSARIO

ADC: Analog-Digital Conversion (Conversión Analógica-Digital).

ANN: Artificial Neural Network (Red Neuronal Artificial).

CNN: Convolutional Neural Network (Red Neuronal Convolucional).

ERPs: Evoked Response Potential. Es una medida de la respuesta cerebral provocada frente a un estímulo.

fMRI: Functional Magnetic Image Resonance. Es un procedimiento que utiliza imágenes de resonancia magnética para medir los pequeños cambios en el flujo sanguíneo que ocurren en una parte activa del cerebro.

HDF5: Hierarchical Data Format version 5.

MFCC: Mel Frequency Cepstral Coefficients (Coeficientes Cepstrales en las Frecuencias de Mel).

MLP: Multilayer Perceptron (Perceptrón multicapa).

MP3: MPEG-1 Audio Layer 3.

RAE: Real Academia Española.

RNN: Recursive Neural Network (Red Neuronal Recurrente).

STFT: Short-time Fourier transform (Transformada de Fourier de Tiempo Reducido).

WAV: Waveform Audio File Format (Formato de archivo de audio de forma de onda).

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	V
GLOSARIO	VI
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS	IX
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	3
1.1 Planteamiento del problema	5
1.2 Objetivos	6
CAPÍTULO 2: MARCO CONCEPTUAL	7
2.1 Métodos de representación de información para ondas sonoras	7
2.1.1 Transformada Discreta de Fourier (DFT)	7
2.1.2 Transformada de Fourier de Tiempo Reducido (STFT) en tiempo discreto	8
2.1.3 Coeficientes Cepstrales en las Frecuencias de Mel (MFCC)	9
2.2 Redes Neuronales Artificiales (ANN)	12
2.2.1 Funciones de activación	13
2.2.2 Perceptrón Multicapa (MLP)	13
2.2.3 Redes Neuronales Convolucionales (CNN)	15
2.2.4 Redes Neuronales Recursivas (RNN)	16
2.3 Metodología	17
2.3.1 Metodología Fundamental para la Ciencia de Datos	18
2.4 Detector de mentiras	20
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	22
3.1 Etapa 1: Comprensión del negocio	22
3.2 Etapa 2: Enfoque analítico	23
3.3 Etapa 3: Requisitos de datos	23
3.4 Etapa 4: Recopilación de datos	24
3.5 Etapa 5: Comprensión de datos	25
3.6 Etapa 6: Preparación de datos	26
3.7 Etapa 7: Modelado	27
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	33
4.0.1 Etapa 8: Evaluación	33
4.0.2 Etapa 9: Implementación	33

4.0.3	Etapa 10: Retroalimentación	35
CAPÍTULO 5: CONCLUSIONES		36
ANEXOS		41
A	Anexo A	41
B	Anexo B	42
C	Anexo C	45
D	Anexo D	51
E	Anexo E	55
F	Anexo F	56
F.1	Procesamiento del sonido	56
F.1.1	Sonido	56
F.1.2	Conversión Analógica-Digital	57
G	Anexo G	58
G.1	Redes Neuronales Artificiales (ANN)	59
H	Anexo H	60
H.1	Aprendizaje y entrenamiento	61
REFERENCIAS BIBLIOGRÁFICAS		63

ÍNDICE DE FIGURAS

1	Árbol de problemas.	5
2	Ejemplo del funcionamiento de una STFT [MathWorks, 2019].	9
3	Ejemplo de MLP y sus componentes.	14
4	Proceso de convolución [Serrano, 2019].	16
5	Proceso de agrupación [Serrano, 2019].	16
6	Una red neuronal recurrente desenrollada [Olah, 2015].	17
7	Diagrama de la metodología fundamental para la ciencia de datos [Rollins, 2015].	18
8	WAV vs MP3 [Fabesgl, 2015].	24
9	Topología MLP.	30
10	Topología CNN.	31
11	Topología RNN.	32
12	Página principal de la aplicación Web para detectar mentiras.	34
13	Ejemplos del gráfico de confianza en función del tiempo en la aplicación web.	35
14	Características de una onda sinusoidal [Jaramillo, 2007].	57
15	Esquema de la ADC [Commons, 2021].	58
16	Cuantificación de una señal de audio [Commons, 2020].	58
17	Esquema de una Neurona [Velardo, 2020].	59
18	Red Neuronal Artificial [Mohanty, 2019].	60

ÍNDICE DE TABLAS

1	Tabla de frecuencia de las variables "Sex"(sexo) y "Veracity"(veracidad).	25
2	Resumen de datos de los videos	26
3	Cantidad del conjunto de datos utilizados en los modelos.	28
4	Dimensión de cada dato y en cada modelo.	28
5	Resultados de las métricas de entrenamiento de cada modelo.	33
6	Resultados de las métricas de entrenamiento de cada modelo.	35

INTRODUCCIÓN

En la era de la información y la comunicación, la detección de mentiras ha adquirido una relevancia significativa en diversos ámbitos, desde la seguridad hasta la psicología y el derecho. Comprender la naturaleza de la mentira y desarrollar métodos confiables para identificarla se ha convertido en un desafío de gran importancia. En este trabajo, se aborda el tema de la detección de mentiras, explorando sus antecedentes históricos y proponiendo una solución basada en técnicas de aprendizaje profundo.

La detección de mentiras ha sido objeto de estudio a lo largo de la historia de la humanidad. Desde los primeros intentos por identificar engaños mediante la observación del lenguaje corporal hasta los modernos enfoques basados en el análisis de datos, el campo ha evolucionado significativamente. En este trabajo, se realiza una revisión exhaustiva de la literatura existente, comenzando por definir la mentira y explorando sus orígenes en la sociedad humana.

Además de comprender la naturaleza de la mentira, se aborda el desafío de analizar los audios de las personas hablando, que constituyen una fuente valiosa de información para la detección de mentiras. Se presenta una revisión detallada de los métodos físicos utilizados para el análisis de audio, centrándose en aquellos que permiten capturar características relevantes para diferenciar la veracidad de las declaraciones.

Para el análisis de los audios y preparación del conjunto de datos de entrenamiento, se emplean diversos métodos matemáticos, incluyendo la transformada de Fourier y los coeficientes cepstrales de Mel. Estas técnicas permiten extraer características acústicas clave de cada audio.

Con el objetivo de lograr la detección de mentiras en audios, se emplean modelos de aprendizaje profundo. Específicamente, se entrenan tres tipos de modelos con el conjunto de datos generado anteriormente: Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN) y Recurrent Neural Network (RNN). Se evalúa el rendimiento de cada modelo y se selecciona el que ofrece los mejores resultados.

Además de la etapa de investigación y desarrollo de modelos, se construye una pequeña aplicación práctica utilizando el framework Django. Esta aplicación permite subir audios y predecir la veracidad de los mismos. Se presenta una descripción detallada de la arquitectura y funcionamiento de la aplicación, resaltando su utilidad potencial en diversos contextos.

En este trabajo, se sigue la metodología fundamental de la ciencia de datos de IBM, que proporciona un enfoque sistemático para el desarrollo de soluciones basadas en datos. Se describen las etapas de recolección y preparación de datos, entrenamiento y evaluación de modelos, y desarrollo de la aplicación final.

Este trabajo esta compuesto principalmente por 5 capítulos:

1. Capítulo 1: Introducción y Objetivos. En este capítulo se plantea y define el problema de la detección de mentiras y se establecen los objetivos y alcance de la memoria.
2. Capítulo 2: Marco Conceptual y Estado del Arte. Se definen los conceptos técnicos, metodologías y herramientas utilizadas, incluyendo un estado del arte con algunas soluciones existentes.
3. Capítulo 3: Propuesta de Solución. Se desarrolla la solución con la metodología fundamental para la ciencia de datos que fue descrita en el capítulo anterior, paso a paso.
4. Capítulo 4: Validación de la Solución. En este capítulo se realiza la validación de la propuesta de solución, analizando los resultados obtenidos en términos de precisión y rendimiento de los modelos, así como las consideraciones y limitaciones encontradas durante el desarrollo del trabajo.
5. Capítulo 5: Conclusiones. Se concluye el trabajo, resumiendo los resultados obtenidos, limitaciones y planteando posibles direcciones futuras de investigación.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

Según la Real Academia Española (RAE), la mentira se define como “expresión o manifestación contraria a lo que se sabe, piensa o siente”. La mentira ha sido un comportamiento que ha acompañado a la humanidad desde tiempos ancestrales, y se ha originado a partir de la necesidad de sobrevivir y prosperar en entornos hostiles. Ha sido observado en diversos momentos históricos, en la naturaleza y en diferentes culturas. De esta manera, la mentira se ha convertido en un aspecto integral de la condición humana y ha evolucionado a lo largo del tiempo.

Por ejemplo, desde los inicios de la humanidad en la prehistoria cuando los hombres primitivos utilizaban distintas técnicas de caza para atrapar a sus presas o cuando se luchaba por su supervivencia en las tribus. También, ha estado presente en diversas culturas, como por ejemplo en la mitología nórdica, donde un personaje llamado Loki es conocido como el dios de las mentiras por sus travesuras; En la mitología griega, donde los inmortales dioses, a pesar de su divinidad y poder, se mienten entre ellos y a las personas; En el cristianismo, en la biblia se relata la historia de una serpiente que persuade a una mujer llamada Eva de comer el fruto de un árbol prohibido que supuestamente una sabiduría y conocimiento divino [Fernández, 2006].

La mentira se manifiesta en las personas a una temprana edad y ésta aparece cuando los niños comienzan a desarrollar un mecanismo de defensa llamado negación [Vels, 1990]. Aproximadamente el 50 % de los niños comienzan a mentir a los 2 años, y este valor aumentan hasta 80 % para los niños mayores de 4 años. Aprender a mentir para un niño es algo muy positivo, incluso es señal de que el niño avanzó al siguiente nivel de razonamiento, pero ¿Es posible detectar sus mentiras? Se realizó un estudio a diversas personas con distintas profesiones a los cuales se les mostraba un niño respondiendo una pregunta, y ellos debían indicar si el niño estaba mintiendo o no. En promedio las personas tenían una precisión del 50 % para detectar mentiras en los niños [Lee, 2016].

En un trabajo del año 2005, se realizaron diversas investigaciones y análisis respecto a la detección de mentiras, relacionadas con la observación de patrones de comportamiento. Los resultados no fueron muy relevantes, la precisión en la detección de mentiras de las personas oscilaba entre 50 % a un 55 % luego de aplicar estas técnicas [Masip, 2005].

La mentira es un comportamiento común que se manifiesta diariamente en nuestra sociedad, principalmente se puede presentar de diferentes formas, como por ejemplo, en falsificaciones, manipulaciones y, en general, en comportamientos engañosos. Éstas se pueden dividir en tres categorías [Martinez, 2021]:

- **Mentiras Blancas:** Son inofensivas, y generalmente no tienen un gran impacto. Tienen el objetivo de causar el menor daño posible, ya que están relacionadas con un fin

moral, una buena intención o un acto altruista.

- **Mentiras Azules:** No son inofensivas, pero tampoco son graves. Su objetivo es proteger al grupo al cual se pertenece, es decir, mentir por el bien de un grupo.
- **Mentiras Negras:** Son graves, y pueden llegar a afectar la vida de las personas. Su objetivo es evadir culpas, escapar de situaciones desfavorables, manipular, entre otros. Son influenciadas por una motivación egoísta.

Como consecuencia, las mentiras pueden llegar a afectar varias instituciones, organizaciones, lugares e incluso sociedades, algunos casos de ejemplos son: En los tribunales de justicia, cuando se libera a un acusado culpable, y por el contrario, cuando se condena a un acusado inocente [Chabria, 2019] [Dollinger, 2019]; En la política, cuando se deben responder preguntas en público [Hinojosa, 2020]; En inversionistas, cuando deben escoger si dar o no financiamiento [Advice, 2020]; Entrevistas de trabajo, cuando se debe decidir si contratar o no a una persona; Entre otros casos.

Dentro de los stakeholders mencionados anteriormente, se encuentra la empresa Digevo [Digevo, 1997] específicamente para el equipo de Digevo Ventures [Ventures, 1997], quienes se encargan de evaluar la viabilidad de distintos proyectos de emprendimiento de personas que postulan al programa de incubación de Digevo. Los beneficios de este programa incluyen recibir asesorías, mentorías e incluso financiamiento para finalmente transformar las ideas en startups. Dentro de los elementos que deben evaluar de los proyectos, se encuentra un video pitch en donde una persona presenta su idea y además debe responder algunas preguntas. Generalmente en el video no se muestra al presentador, por lo que en algunos videos no es posible analizar patrones no verbales, por lo tanto, los evaluadores deben determinar si la información declarada por la persona es verdadera o falsa solo analizando el audio, razón por la cual nace la idea de este proyecto.

En la figura 1 se puede apreciar un árbol de problemas elaborado con la ayuda de los antecedentes presentados anteriormente.

A continuación, se presenta el árbol de problemas:

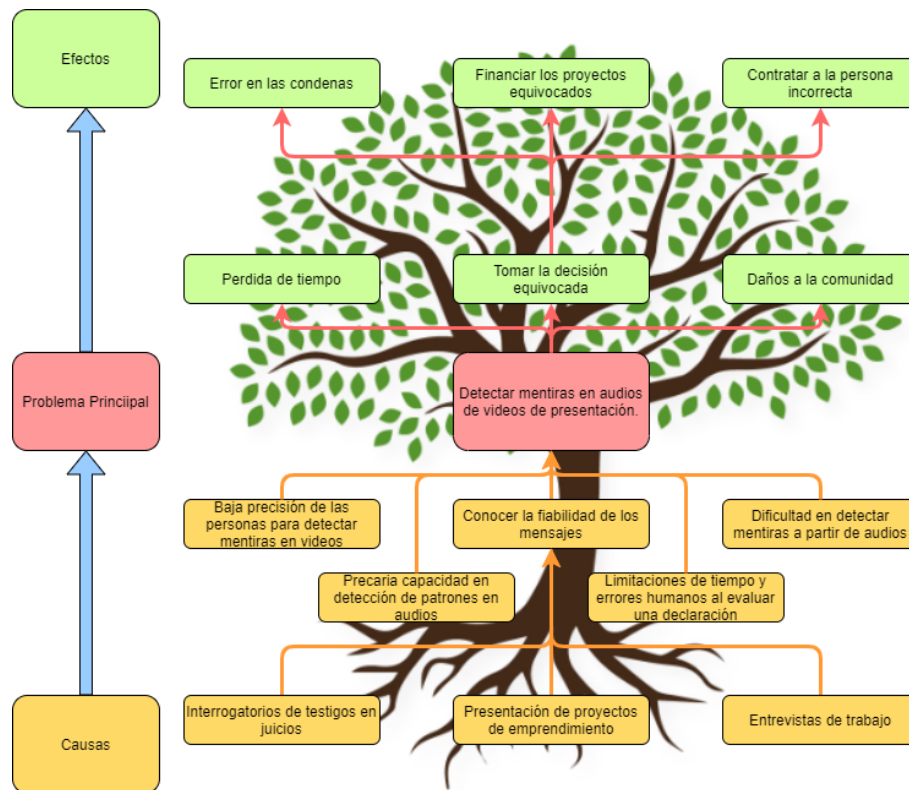


Figura 1: Árbol de problemas.

1.1. Planteamiento del problema

Mentir, dentro del contexto de este trabajo, es manifestar una declaración falsa con el objetivo de hacer que otras personas creen que es verdad; No obstante, cuando una persona miente experimenta una serie de sentimientos y emociones que se reflejan físicamente en su cuerpo, y que se pueden percibir en sus expresiones faciales, voz, pulso, flujo sanguíneo en su cara, etc. Estos sentimientos y emociones que experimentan las personas forman patrones comunes tanto para las declaraciones falsas como para las verdaderas. Estos patrones pueden ser detectados y existen estudios, como uno realizado en 1989 [Becerra *et al.*, 1989], que sugieren que las personas son levemente capaces de detectar estos patrones, después de observarlos en videos.

En la actualidad existen muchas herramientas y técnicas para detectar mentiras, como por ejemplo: el polígrafo, ERPs, fMRI, el análisis de expresiones faciales, análisis y detección del flujo sanguíneo en el rostro, etc. Sin embargo, herramientas como el polígrafo, el ERPs y el fMRI [Masip, 2017], las cuales alcanzan una precisión que ronda entre el 70 – 80 %, se basan principalmente en analizar el estado cerebral de una persona a medida que se le hacen preguntas, por lo que estas herramientas requieren de maquinarias muy costosas y además de personas expertas para poder utilizarlas e interpretar los resultados que se generan.

A continuación, se presentarán el objetivo general y los objetivos específicos de este trabajo.

1.2. Objetivos

El objetivo general del presente trabajo es:

- Aplicar e integrar una metodología que permita detectar mentiras a partir del audio de videos de presentación con Machine Learning.

Este objetivo general se logra a través de los siguientes objetivos específicos:

- Analizar transformadas matemáticas que permitan obtener una representación apropiada de las señales de onda de los audios.
- Buscar y comparar diversos modelos de Machine Learning que permitan trabajar con audios.
- Evaluar el modelo Machine Learning propuesto para la detección de mentiras en audios.

CAPÍTULO 2

MARCO CONCEPTUAL

“Entender una pregunta es tener ya media respuesta”
(Sócrates)

El objetivo de este capítulo es describir las bases conceptuales, métodos y fundamentos técnicos que están involucradas en la solución propuesta. Primero se definirá algunos conceptos básicos sobre el sonido y su digitalización. Luego se describirán algunos métodos populares que se han utilizado para extraer información de ondas sonoras. Por último, se describirán brevemente algunos modelos de Deep Learning de interés trabajados en esta memoria y una metodología para la ciencia de datos.

2.1. Métodos de representación de información para ondas sonoras

2.1.1. Transformada Discreta de Fourier (DFT)

La transformada Discreta de Fourier, de aquí en adelante DFT, es una transformación matemática que permite realizar un análisis en frecuencia de una señal en forma discreta. [Albiol *et al.*, 1998]

Matemáticamente la DFT se define de la siguiente forma:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}} \quad k = 0, \dots, N-1 \quad (1)$$

Donde:

- N es la cantidad de datos de $x[n]$.
- $x[n]$ es la señal discreta, es decir, la secuencia de duración limitada de N datos.

$$x[n] = \begin{cases} x_1, x_2, \dots, x_N & 1 \leq n \leq N \\ 0 & \text{otro caso} \end{cases} \quad (2)$$

2.1.2. Transformada de Fourier de Tiempo Reducido (STFT) en tiempo discreto

La transformada de Fourier de Tiempo Reducido (STFT) en tiempo discreto, es una transformación matemática que permite realizar análisis de tiempo y frecuencia de una señal en forma discreta. [Albiol *et al.*, 1998]

Matemáticamente la STFT discreta se define de la siguiente forma:

$$STFT \{w[m]\} \equiv X[n, \omega_k] = \sum_{m=0}^{N-1} w[m] x[n+m] e^{-j\omega_k m} \quad (3)$$

Donde:

- N es la cantidad de datos de $x[n]$.

-

$$\omega_k = \frac{2\pi k}{N} \quad (4)$$

- $x[n+m]$ es la señal discreta desplazada m datos hacia la derecha.
- $w[m]$ es la función ventana que permite evitar discontinuidades al principio y al final de los bloques analizados.

$$w[m] = \begin{cases} h(t) & t \in [T_{0+m}, T_{n+m}] \\ 0 & \text{otro caso} \end{cases} \quad (5)$$

Notar que cada ventana o bloque de tiempo de la STFT discreta corresponde a calcular una DFT.

En la figura 2 se puede visualizar un gráfico ejemplo de una stft.

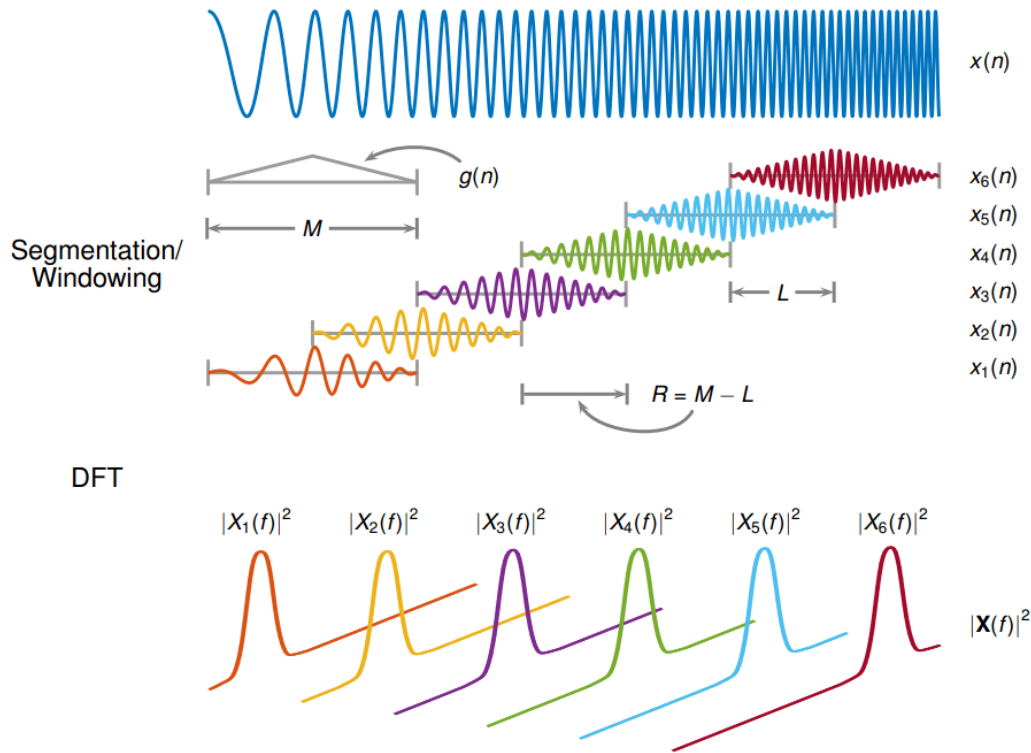


Figura 2: Ejemplo del funcionamiento de una STFT [MathWorks, 2019].

2.1.3. Coeficientes Cepstrales en las Frecuencias de Mel (MFCC)

Los coeficientes Cepstrales en las Frecuencias de Mel o MFCC [Aizawa *et al.*, 2004] son una técnica utilizada en el campo del procesamiento de señales y reconocimiento de voz para representar y analizar el contenido de frecuencia de una señal de audio. Esta técnica se basa en la transformada cepstral, que es una transformada matemática que se utiliza para analizar la estructura de frecuencia de una señal de audio.

La idea detrás de los coeficientes cepstrales en las frecuencias de Mel es mapear la energía de una señal de audio en una escala de frecuencias que es más parecida a la forma en que nuestro oído percibe el sonido. Esto se logra mediante el uso de la escala de Mel, que es una escala logarítmica que se basa en la forma en que nuestro oído percibe las diferencias de frecuencia en el sonido.

Una vez que se ha mapeado la señal de audio en la escala de Mel, se puede aplicar la transformada cepstral para obtener los coeficientes cepstrales en las frecuencias de Mel. Estos coeficientes se pueden utilizar para representar y analizar la estructura de frecuencia de la señal de audio, lo que resulta útil en aplicaciones como el reconocimiento de voz y la identificación de patrones en señales de audio.

Cabe señalar que el cálculo de los coeficientes cepstrales en las frecuencias de mel puede variar ligeramente dependiendo del enfoque utilizado, pero en general se siguen los pasos descritos anteriormente. A continuación un ejemplo con mayor detalle. [Giannakopoulos y Pirkakis, 2014]

1. Aplicar una ventana de Fourier a la señal de audio, lo que permite dividir la señal en una serie de segmentos de tiempo. Esto se puede escribir matemáticamente como:

$$s[n] \xrightarrow{\text{ventana de Fourier}} s_w[n] = s[n] \cdot w[n] \quad (6)$$

donde $s[n]$ es la señal de audio en el tiempo discreto n , $s_w[n]$ es la señal de audio ventaneada en el tiempo discreto n , y $w[n]$ es la función de ventana de Fourier.

2. Calcular la transformada discreta de Fourier (DFT) de cada uno de los segmentos de tiempo obtenidos en el primer paso. Esto permite obtener la representación de frecuencia de cada segmento de tiempo. Esto se puede escribir matemáticamente como:

$$S_w[k] = \sum_{n=0}^{N-1} s_w[n] \cdot e^{-j\frac{2\pi}{N}kn} \quad (7)$$

donde $S_w[k]$ es la DFT de la señal de audio ventaneada en la frecuencia discreta k , y N es el número de muestras en cada segmento de tiempo.

3. Aplicar un filtro de Mel a cada una de las representaciones de frecuencia obtenidas en el segundo paso. Esto permite mapear la energía de la señal de audio en la escala de Mel. Esto se puede escribir matemáticamente como:

$$S_{w,m}[k] = S_w[k] \cdot H_m[k] \quad (8)$$

donde $S_{w,m}[k]$ es la señal de audio ventaneada y filtrada en la escala de Mel en la frecuencia discreta k , y $H_m[k]$ es el filtro de Mel en la frecuencia discreta k .

4. Tomar el logaritmo de las representaciones de frecuencia filtradas obtenidas en el tercer paso. Esto se puede escribir matemáticamente como:

$$S_{w,m,\log}[k] = \log(S_{w,m}[k]) \quad (9)$$

donde $S_{w,m,\log}[k]$ es la señal de audio ventaneada, filtrada y logaritmizada en la escala de Mel en la frecuencia discreta k .

5. Calcular la transformada cepstral de cada una de las representaciones de frecuencia filtradas y logaritmizadas obtenidas en las etapas anteriores. Esto se puede hacer mediante una transformada de coseno discreta inversa (IDCT, por sus siglas en inglés).

Esto permite obtener los coeficientes cepstrales en las frecuencias de Mel de la señal de audio. Esto se puede escribir matemáticamente como:

$$C[k] = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} S_{w,m,\log}[k] \cdot \cos\left(\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right) \quad (10)$$

donde $C[k]$ es la transformada cepstral de la señal de audio en la frecuencia discreta k , y N es el número de muestras en cada segmento de tiempo.

Para calcular $H_m[k]$ del punto 3, que es el filtro de Mel en la frecuencia discreta k [Proakis y Manolakis, 1996], se pueden seguir los siguientes pasos:

1. Definir un rango de frecuencias en la escala de Mel que se quiere utilizar. Esto se puede hacer mediante una función que convierta las frecuencias en Hertz a la escala de Mel, por ejemplo:

$$f_{m,min} = 2595 \log_{10} \left(1 + \frac{f_{min}}{700} \right) \quad (11)$$

$$f_{m,max} = 2595 \log_{10} \left(1 + \frac{f_{max}}{700} \right) \quad (12)$$

donde $f_{m,min}$ y $f_{m,max}$ son los límites inferiores y superiores, respectivamente, del rango de frecuencias en la escala de Mel, y f_{min} y f_{max} son los límites inferiores y superiores, respectivamente, del rango de frecuencias en Hertz.

2. Dividir el rango de frecuencias en la escala de Mel en un número finito de bandas de igual ancho. Esto se puede hacer mediante una función que calcule los centros de las bandas en la escala de Mel, por ejemplo:

$$f_m[k] = f_{m,min} + k \cdot \frac{f_{m,max} - f_{m,min}}{K} \quad (13)$$

donde $f_m[k]$ es el centro de la banda k en la escala de Mel, K es el número de bandas, y k varía de 0 a $K - 1$.

3. Generar una función de ventana de Fourier para cada una de las bandas definidas en el segundo paso. Esto se puede hacer mediante una función que calcule los valores de la ventana de Fourier en la escala de Mel, por ejemplo:

$$w_m[k] = \begin{cases} 1 - \frac{1}{2} \left| \frac{k - f_{m,1}}{f_{m,2} - f_{m,1}} \right| & \text{si } f_{m,1} \leq k \leq f_{m,2} \\ 1 - \frac{1}{2} \left| \frac{k - f_{m,3}}{f_{m,4} - f_{m,3}} \right| & \text{si } f_{m,3} \leq k \leq f_{m,4} \\ 0 & \text{en otro caso} \end{cases} \quad (14)$$

donde $w_m[k]$ es el valor de la ventana de Fourier en la frecuencia discreta k en la escala de Mel, y $f_{m,1}$, $f_{m,2}$, $f_{m,3}$ y $f_{m,4}$ son las frecuencias en la escala de Mel que delimitan las bandas de la ventana de Fourier.

4. Convolucionar las funciones de ventana de Fourier generadas en el tercer paso para obtener el filtro de Mel en la escala de Mel. Esto se puede escribir matemáticamente como:

$$H_m[k] = \sum_{i=0}^{K-1} w_m[i] \cdot \delta[k - i] \quad (15)$$

donde $H_m[k]$ es el filtro de Mel en la frecuencia discreta k en la escala de Mel, $w_m[i]$ es el valor de la ventana de Fourier en la frecuencia discreta i en la escala de Mel, K es el número de bandas, y $\delta[k]$ es la función delta de Dirac en la frecuencia discreta k .

La función delta de Dirac es una función matemática utilizada en el análisis de señales y sistemas. Matemáticamente, se define como:

$$\delta[k] = \begin{cases} 0 & \text{si } k \neq 0 \\ 1 & \text{si } k = 0 \end{cases} \quad (16)$$

Por último, la función delta de Dirac es una función muy útil en el cálculo de la convolución entre dos señales. La convolución se define como el producto de una señal por una función de retraso, seguido de una integración sobre el tiempo. Si una de las señales es la función delta de Dirac, entonces el resultado de la convolución es simplemente la otra señal retrasada en el tiempo, es decir:

$$f[k] * \delta[k] = f[k - n] \quad (17)$$

donde $f[k]$ es la señal original, n es el retraso en el tiempo, y $*$ representa la operación de convolución.

En el caso del procedimiento para calcular los coeficientes cepstrales en las frecuencias de Mel de una señal de audio, la función delta de Dirac se utiliza en el cuarto paso para calcular el filtro de Mel en la escala de Mel. Esto se hace mediante la convolución de las funciones de ventana de Fourier generadas en el tercer paso, que permite obtener un filtro de Mel que se ajusta al ancho de las bandas de frecuencia definidas en la escala de Mel.

2.2. Redes Neuronales Artificiales (ANN)

Las Redes Neuronales Artificiales, o ANN (Artificial Neural Network), son un conjunto de algoritmos creados con el objetivo de emular el funcionamiento sistema nervioso del cerebro humano [McCulloch y Pitts, 1943]. Estas redes suelen estar compuestas por capas de neuronas, pesos y funciones de activación. En el anexo G se puede encontrar información más detallada sobre las ANN.

2.2.1. Funciones de activación

Las funciones de activación son una parte esencial de las redes neuronales artificiales [Nielsen, 2015a]. Se utilizan para introducir no linealidad en los modelos de redes neuronales, lo que les permite aprender y representar una amplia variedad de funciones y patrones complejos en los datos [Goodfellow *et al.*, 2016].

Matemáticamente, una función de activación se puede definir como una función matemática $f(x)$ que toma una entrada x y produce una salida y según la siguiente ecuación:

$$y = f(x) \quad (18)$$

Hay varias funciones de activación diferentes que se utilizan en redes neuronales, cada una con sus propias características y aplicaciones [Goodfellow *et al.*, 2016]. Algunas de las funciones de activación más comunes son la función sigmoide, la función tangente hiperbólica y la función ReLU [Nair y Hinton, 2010].

La función sigmoide, por ejemplo, se define como:

$$f(x) = \frac{1}{1 + e^{-x}} \quad (19)$$

mientras que la función tangente hiperbólica se define como:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (20)$$

La función ReLU (rectified linear unit), por su parte, se define como:

$$f(x) = \max(0, x) \quad (21)$$

En general, la elección de una función de activación depende del tipo de problema que se esté tratando y del diseño de la red neuronal en particular [Goodfellow *et al.*, 2016]. Por lo tanto, es importante entender las propiedades y características de cada función de activación para poder utilizarlas de manera efectiva en el diseño y entrenamiento de redes neuronales.

2.2.2. Perceptrón Multicapa (MLP)

Un Perceptrón Multicapa o MLP es un tipo de ANN, el cual está compuesto principalmente por: Una red de perceptrones; Capas de entrada, ocultas y de salida; pesos de conexión; y

por último, funciones de activación. Un perceptrón es un tipo de neurona artificial que fue desarrollado entre 1950 y 1960 por el científico Frank Rosenblatt [Nielsen, 2015b] y es uno de los primeros ANN desarrollados con la capacidad de modelar funciones altamente no lineales.

En la figura 3 se puede visualizar un ejemplo de los componentes que posee un MLP.

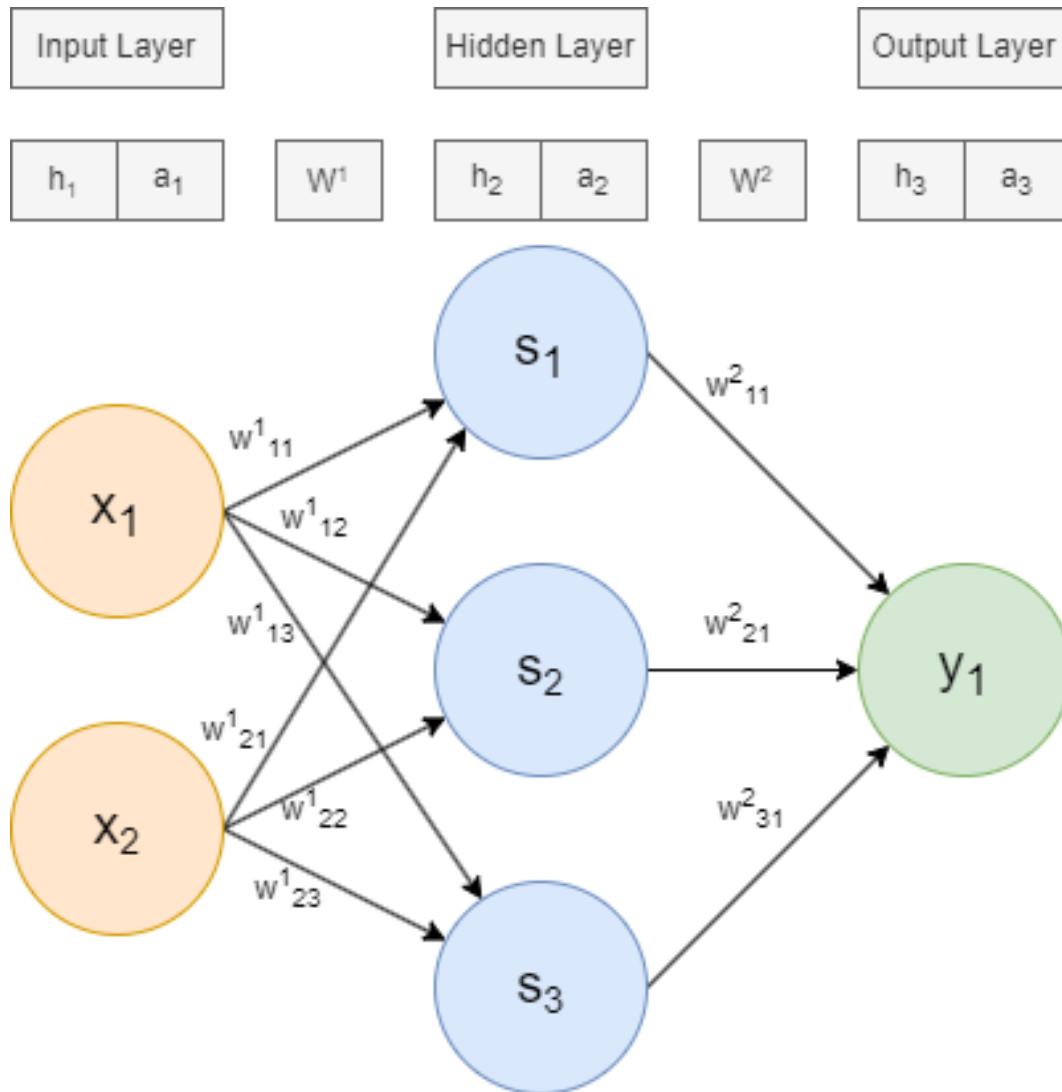


Figura 3: Ejemplo de MLP y sus componentes.

Estos componentes se pueden definir de la siguiente manera;

- **Input Layer:** En la capa de entrada $h_1 = a_1$ y son los datos dados a la MLP.

- **Hidden Layer:** En la capa oculta h_2 y a_2 se pueden calcular de la siguiente forma:

$$h_2 = xW^1 = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} w_{11}^1 & w_{12}^1 & w_{13}^1 \\ w_{21}^1 & w_{22}^1 & w_{23}^1 \end{bmatrix} \quad (22)$$

$$a_2 = f(h_2) \quad (23)$$

- **Output Layer:** En la capa de salida h_3 y a_3 se pueden calcular de la misma forma que h_2 y a_2 , pero utilizando los datos de salida de la capa oculta como input, es decir:

$$h_3 = a_2W^2 = \begin{bmatrix} s_1 & s_2 & s_3 \end{bmatrix} \begin{bmatrix} w_{21}^2 \\ w_{22}^2 \\ w_{23}^2 \end{bmatrix} \quad (24)$$

$$y_1 = a_3 = f(h_3) \quad (25)$$

El siguiente paso es encontrar y ajustar los pesos W del modelo MLP para que pueda desempeñar su tarea correctamente, este proceso se denomina aprendizaje. Para esto se necesitaran dos componentes fundamentales, un conjunto de datos de entrenamiento y una función de perdido o error [Nielsen, 2015b].

2.2.3. Redes Neuronales Convolucionales (CNN)

Las Redes Neuronales Convolucionales o CNN, son un tipo de ANN, los cuales están basados en el mecanismo de percepción visual de los mamíferos, específicamente en las células simples y complejas que procesan las imágenes en la corteza visual, y que ayudan a detectar características en una imagen. Las CNNs poseen un conjunto de capas convolucionales y un conjunto de capas de agrupación, las cuales tienen el objetivo de imitar el comportamiento de las células simples y complejas respectivamente, permitiendo a una CNN aprender progresivamente e identificar diversas características desde una imagen. [LeCun *et al.*, 2015]

La capa de convolución [Nielsen, 2015a] permite generar una gran variedad de imágenes a partir de la imagen de entrada que recibe la capa (inputs) y todas del mismo tamaño. Para generar las imágenes, esta capa realiza un proceso de convolución el cual consiste en tomar cada pixel de la imagen y las de alrededor para luego multiplicarlas por una matriz, denominada Kernel, la cual permite extraer ciertas características de la imagen según los valores que contenga.

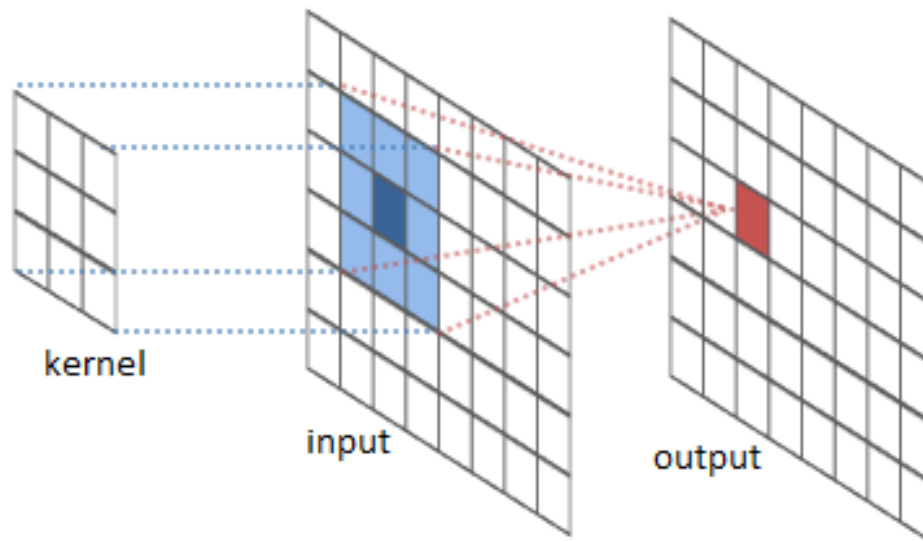


Figura 4: Proceso de convolución [Serrano, 2019].

La capa de agrupación [Nielsen, 2015a] permite generar una imagen más pequeña que la imagen de entrada que recibe la capa. El objetivo de generar esta imagen más pequeña es conseguir una mayor abstracción sobre los datos. Generalmente se utiliza un proceso de agrupación llamado “Max pooling”, el cual consiste en agrupar la imagen en pequeñas matrices de tamaños definidos, para posteriormente almacenar el mayor valor de cada y generar un nuevo conjunto de datos, tal y como se muestra en la imagen 5.

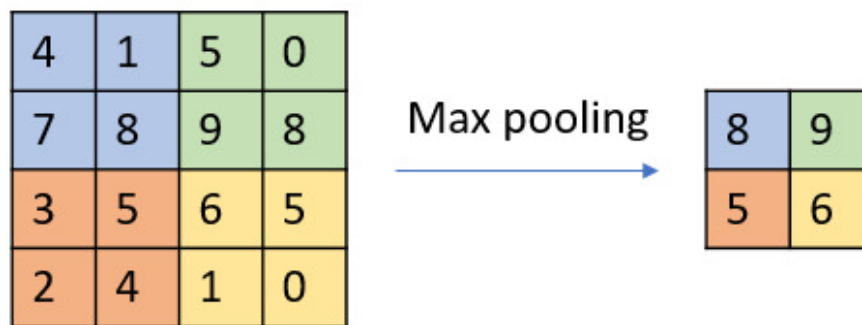


Figura 5: Proceso de agrupación [Serrano, 2019].

2.2.4. Redes Neuronales Recursivas (RNN)

Las Redes Neuronales Recursivas o RNN, son un tipo de ANN, las cuales tienen el objetivo de imitar el razonamiento del cerebro sobre una sucesión de eventos. Las RNN son siste-

mas dinámicos en donde, el cálculo de una entrada depende del paso anterior. Estas redes contienen bucles en ellas permitiendo la que información persista. [Olah, 2015]

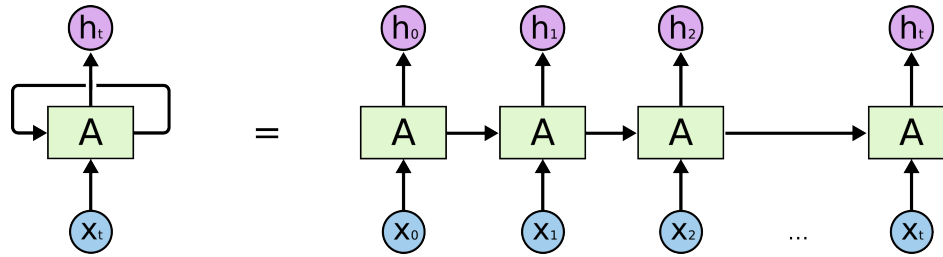


Figura 6: Una red neuronal recurrente desenrollada [Olah, 2015].

Las RNN son capaces de conectar información previa con la tarea actual, sin embargo, a medida que necesitan información relevante de mucho más atrás, se le comienza a dificultar la conexión de la información. Afortunadamente en 1997 Hochreiter y Schmidhuber crearon un tipo de RNN llamado Long Short Term Memory networks o también conocido como RNN-LSTM, este tipo de RNN tuvo mucho éxito en una variedad de problemas, como por ejemplo: reconocimiento de voz, modelado de lenguaje, traducción, subtítulos de imágenes, etc.

2.3. Metodología

La metodología es un conjunto de principios y técnicas que se utilizan para abordar y resolver problemas de manera lógica y sistemática. En el contexto de la ciencia de datos, se puede decir que la metodología fundamental es el conjunto de pasos o etapas que se siguen para llevar a cabo un proyecto de análisis de datos de manera efectiva y organizada [Provost y Fawcett, 2013].

La metodología de la ciencia de datos se compone de cuatro etapas principales:

1. **Recolección de datos:** en esta etapa se busca obtener los datos que se van a utilizar para el análisis, ya sea a través de la recopilación de datos nuevos o mediante el uso de datos existentes.
2. **Limpieza y preparación de datos:** en esta etapa se limpian y preparan los datos para su análisis, lo que incluye tareas como la eliminación de datos incompletos o inconsistentes, la corrección de errores y la transformación de los datos en un formato adecuado para su análisis.
3. **Análisis de datos:** en esta etapa se aplican técnicas de análisis de datos, como el análisis estadístico o el aprendizaje automático, para obtener información valiosa a partir de los datos.

4. Presentación de resultados: en esta etapa se presentan los resultados del análisis de datos de manera clara y concisa, ya sea en forma de informes, gráficos o visualizaciones de datos.

Es importante señalar que estas cuatro etapas no se llevan a cabo de manera secuencial, sino que se pueden iterar y volver a realizar varias veces hasta obtener resultados satisfactorios. Además, esta metodología puede variar ligeramente según el contexto y el objetivo del proyecto de análisis de datos. Sin embargo, en general, se puede decir que estas cuatro etapas son la base de la metodología de la ciencia de datos.

2.3.1. Metodología Fundamental para la Ciencia de Datos

La metodología fundamental para la ciencia de datos de IBM [Rollins, 2015] es una estrategia de orientación para el análisis de datos y la resolución de problemas a través de la construcción de modelos para predecir resultados o descubrir patrones subyacentes. La metodología consta de 10 etapas que forman un proceso iterativo y se enfoca en el uso de grandes volúmenes de datos, la incorporación de la analítica de texto en el modelado predictivo y la automatización de algunos procesos. Estas etapas se pueden ver en la figura 7.

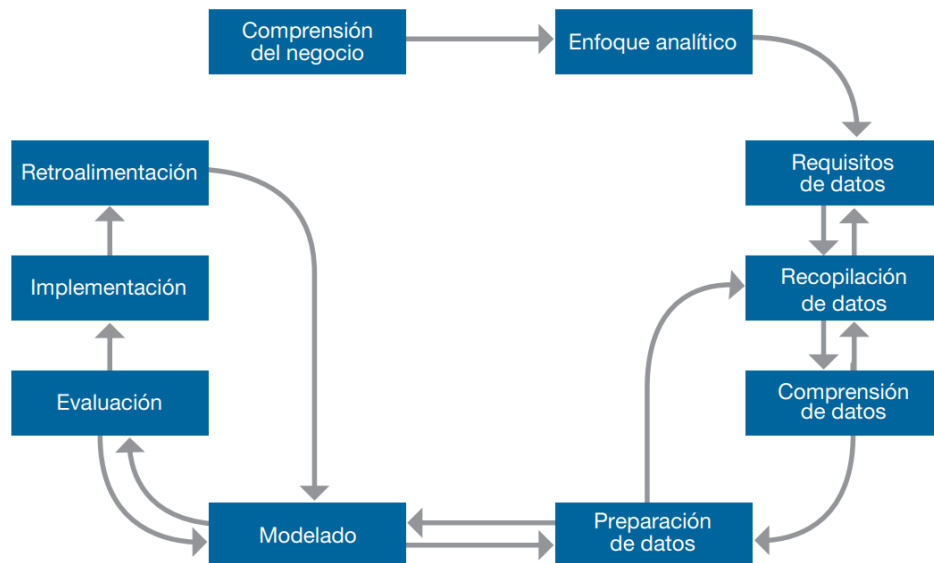


Figura 7: Diagrama de la metodología fundamental para la ciencia de datos [Rollins, 2015].

La metodología fundamental para la ciencia de datos de IBM, consta de 10 etapas que forman un proceso iterativo, para el uso de datos para descubrir insights. Cada etapa juega un papel vital en el contexto de la metodología general, y es necesario completar cada una de ellas para resolver problemas y responder preguntas a través del análisis de datos.

1. **Comprensión del negocio:** Definir el problema, los objetivos del proyecto y los requisitos de la solución desde una perspectiva empresarial. Esta primera etapa sienta las bases, para que el problema empresarial sea resuelto con éxito.
2. **Enfoque analítico:** Definir el enfoque analítico para resolver el problema. Esta etapa implica expresar el problema bajo el contexto de las técnicas estadísticas, y de aprendizaje automático, para que la organización pueda identificar las más adecuadas para el resultado deseado.
3. **Requisitos de datos:** Determinar qué tipo y cantidad de datos se necesitan para el enfoque analítico elegido. Esta etapa implica identificar qué datos son relevantes para el problema, y cuál es el formato adecuado para utilizarlos en el análisis.
4. **Recopilación de datos:** Identificar y reunir los recursos de datos disponibles y relevantes para el dominio del problema. Pueden incluir datos estructurados, no estructurados y semiestructurados, y pueden elegir si realizan inversiones adicionales para obtener datos menos accesibles.
5. **Comprensión de datos:** Examinar los datos recopilados para entender su contenido, su formato y su distribución. Se pueden utilizar técnicas como la visualización de datos y el cálculo de estadísticas básicas para identificar patrones y tendencias en los datos, y también para detectar posibles problemas o errores en los datos. La comprensión de los datos es importante para asegurarse de que los datos estén listos para el análisis y el modelado.
6. **Preparación de datos:** Limpiar, transformar y preparar los datos para su uso en el análisis y modelado. Esta etapa implica eliminar datos incompletos, inconsistentes o irrelevantes, y transformar los datos en un formato adecuado para su uso en el análisis. Además, también puede incluir técnicas como la normalización y la agregación de datos para prepararlos para el análisis posterior. La preparación adecuada de los datos es clave para asegurarse de que el análisis y modelado se realicen de manera efectiva y produzcan resultados precisos.
7. **Modelado:** Utilizar técnicas estadísticas y de aprendizaje automático para construir un modelo que represente los patrones, y tendencias en los datos. Este modelo se utiliza para hacer predicciones y extraer información valiosa de los datos. La etapa de modelado es muy importante, ya que, el modelo creado en esta etapa se utilizará para obtener insights y hacer predicciones precisas.
8. **Evaluación:** Validar la calidad y la precisión del modelo creado en la etapa anterior. Esta etapa implica evaluar el modelo en conjuntos de datos distintos al utilizado en la etapa de modelado, con el fin de comprobar si el modelo es generalizable, y tiene un rendimiento aceptable en situaciones distintas a las que se utilizó para su entrenamiento. Esta etapa es importante, ya que, permite evaluar la viabilidad y el rendimiento del modelo en situaciones reales, y determinar si es adecuado para su uso en la resolución del problema empresarial.

9. **Implementación:** Poner en práctica el modelo creado en las etapas anteriores. Esto implica desplegar el modelo en un entorno de producción, y monitorizar su rendimiento para verificar su eficacia y hacer cualquier ajuste necesario. Además, esta etapa incluye la documentación del proceso, y la comunicación de los resultados obtenidos a los stakeholders interesados. Es importante que el modelo implementado sea fácilmente utilizable por los usuarios finales, y que sea escalable y mantenible en el tiempo.
10. **Retroalimentación:** Se centra en la mejora continua del modelo y del proceso a través del tiempo. Esta etapa implica seguir el rendimiento del modelo en el mundo real, y recopilar datos de su uso en la práctica, con el fin de identificar posibles áreas de mejora. Por ejemplo, si se observa que el modelo no está dando buenos resultados en ciertas situaciones, se puede volver a la etapa 7 (Modelado) para mejorar el modelo y volver a evaluarlo en la etapa 8 (Evaluación). También implica revisar periódicamente los objetivos del proyecto y los requisitos de la solución, para asegurarse de que el modelo sigue siendo relevante y adecuado para resolver el problema empresarial.

2.4. Detector de mentiras

En un trabajo de San Agustín (1954) se menciona una de las primeras definiciones de lo que es una mentira: “Mentir es decir lo contrario de lo que uno piensa, con la intención de engañar” [Vivar, 2003].

En un trabajo [Fernández, 2006] se menciona que la mentira ha estado desde los inicios de la humanidad, en su historia y en distintas culturas. Un trabajo [Vels, 1990] analizaba cómo la mentira se manifestaba como el resultado de un mecanismo de defensa llamado negación.

[Masip, 2005] recapitula y analiza distintas técnicas no verbales estudiadas para la detección de mentiras, mostraba que la precisión de las personas para detectar mentiras, antes y después de enseñar estas técnicas, oscilaba entre un 50 % a un 55 %.

La utilización de una herramienta capaz de detectar mentiras por medio de imágenes térmica se puso a prueba en un aeropuerto en donde se entrevistó a 51 pasajeros para preguntarles sobre su próximo viaje [Warmelink *et al.*, 2011]. Esta entrevista fue grabada con una cámara térmica para obtener los datos de la temperatura emanada por las personas. Se clasificó correctamente al 64 % de las personas que dicen la verdad y el 69 % de las personas que dicen mentiras.

[Anta, 2012] menciona las ventajas y desventajas de dos técnicas para el análisis de mentiras el polígrafo y el análisis del lenguaje verbal-corporal. Cuando una persona dice o hace algo que es contrario a lo que cree cierto, se puede entender como una mentira o engaño. Esta acción de mentir se puede manifestar en el cuerpo de manera verbal y no verbal, por lo que instrumentos como el polígrafo, pueden detectarla. Sin embargo, se debe entender que el polígrafo no detecta mentiras o verdades, lo que realmente mide son alteraciones fisiológi-

cas de las personas, como por ejemplo la presión arterial, el ritmo cardiaco, la respiración, etc. Por lo tanto, acciones como morderse la lengua o el labio podrían alterar la prueba del polígrafo. El análisis del lenguaje verbal-corporal se hace por medio de una serie de entrevistas a una persona, en donde principalmente se analiza la relación entre la comunicación verbal y la corporal.

[Gogate *et al.*, 2017] fusionaron características de audio, visuales y textuales para la detección de mentiras en videos. Llegó a 96 % de precisión utilizando Deep Learning de fusión multimodal. Para la extracción de características visuales se utilizó una 3D-CNN, para las características del texto se utilizó text-CNN y, por último, para las características del audio se utilizó openSMILE y luego se utilizó un MLP-classifier.

[Krishnamurthy *et al.*, 2018] llegaron a la cifra de 96,14 % de precisión utilizando el mismo enfoque multimodal mencionado anteriormente, pero además de analizar características visuales, de texto y auditivas, agregaron las características de micro expresiones.

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

Para comenzar con este trabajo de investigación, se buscó y escogió un método que permitiera abordar y proponer una solución a este tipo de problemas. La metodología fundamental para la ciencia de datos, definida en la sección 2, es una técnica de 10 etapas utilizado para trabajar de una manera eficiente con los problemas de ciencias de datos. A continuación, se explicara cada uno de los pasos aplicados a este proyecto de investigación sobre la detección de mentiras con Machine Learning a partir del audio de videos.

3.1. Etapa 1: Comprensión del negocio

La primera etapa consiste principalmente en entender el negocio para definir ciertos elementos clave como el problema del negocio, sus objetivos, los criterios de éxito y los requisitos de solución del negocio.

- **Definición del problema:** Detectar mentiras en las declaraciones de una persona mediante el análisis de voz.
- **Objetivos del negocio:**
 - Detectar mentiras en las declaraciones de las personas a partir del audio.
- **Criterios de éxito del negocio:** Criterios que se necesitan cumplir para resolver el problema.
 - Crear y entrenar un modelo de Deep Learning que permita detectar mentiras y realizar predicciones lo más certeras posibles a partir de un audio.
 - Programar una aplicación web que permita detectar mentiras a partir de un archivo de audio que cargue el usuario.
- **Requisitos de solución:** A continuación se presenta un listado de recursos utilizados para asegurar el cumplimiento del objetivo y los criterios del negocio.
 - Conjunto de datos de declaraciones de personas catalogados en verdad o mentira.
 - Lenguaje de programación Python en donde se utilizarán librerías como NumPy, Librosa, Sklearn, matplotlib y tensor Flow los cuales permitirán analizar y preparar los datos para la creación y entrenamiento de 3 modelos de Deep Learning (MLP, CNN y RNN). Para la parte web, se utilizará la librería Django de Python y la librería tensor Flow de JavaScript para cargar el modelo en la web.

3.2. Etapa 2: Enfoque analítico

La segunda etapa consiste principalmente en definir la cantidad de fases que tendrá el proyecto y que se hará en cada una de ellas.

1. Recopilar distintos conjuntos de datos de videos catalogados
2. Preparación del conjunto de datos a utilizar (selección, limpieza, conversión y formato)
3. Documentar y analizar transformaciones matemáticas que permitan trabajar con señales de onda
4. Extraer y definir los datos relevantes a utilizar de las señales de onda
5. Buscar y analizar posibles modelos candidatos
6. Documentar, programar y entrenar los modelos propuestos
7. Análisis de los resultados obtenidos con los modelos
8. Elaboración de informes con los resultados obtenidos
9. Crear aplicación web con el modelo con mejores resultados y correcciones finales

3.3. Etapa 3: Requisitos de datos

En esta etapa se debe definir los requisitos de los datos. En primer lugar, se debe recopilar un conjunto de datos, ya sea de videos o audios, de personas realizando declaraciones (hablando). Estos datos deben estar catalogados en declaraciones de verdad y declaraciones de mentira. También se debe tener en consideración que el formato final del conjunto de datos debe ser en formato WAV o OGG, ya que la librería “librosa” de Python necesita un formato de audio sin compresión de datos para cargar los datos.

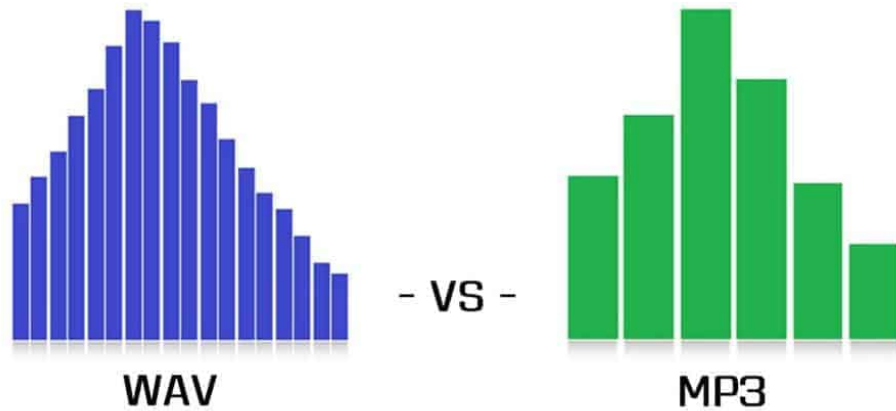


Figura 8: WAV vs MP3 [Fabesgl, 2015].

Como se puede ver en la figura 8, una de las principales diferencias entre el formato WAV y el formato MP3 es la cantidad de datos que se almacenan en cada tipo de archivo. En el formato WAV se guarda una representación exacta de los datos de audio capturados, por otro lado, el formato MP3 toma los datos de audio capturados y luego utiliza algoritmos de compresión para eliminar información considerada como no esencial para el oído humano. [Fabesgl, 2015]

3.4. Etapa 4: Recopilación de datos

En esta etapa se deben recolectar los datos a utilizar, para esto se puede escoger entre dos posibles opciones disponibles. La primera opción consiste en generar un conjunto de datos y catalogarlos, esto implica reunir a un determinado número de personas y realizar una entrevista a cada una de ellas en una habitación la cual tendría preparado un pequeño equipo de grabación y luego ir con expertos a catalogar los datos. La segunda opción consiste en buscar un conjunto de datos ya catalogados por expertos y solicitar su uso para este trabajo. En esta ocasión se optará por la segunda opción.

En el departamento de psicología de la universidad de Denver (University of Denver) crearon en el 2017 una base de datos de videos sobre la detección de mentiras titulado “Miami university deception detection database” (MU3D), que incluye un conjunto de datos de 320 videos de personas hablando en forma honesta y deshonesta sobre sus relaciones sociales [Hugenberg *et al.*, 2017].

Se facilitó acceso al MU3D, que tiene:

- 320 videos.
 - 80 videos con declaraciones positivas verdaderas.

- 80 videos con declaraciones positivas falsas.
 - 80 videos con declaraciones negativas verdaderas.
 - 80 videos con declaraciones negativas falsas.
- Formato video: “.wmv”

3.5. Etapa 5: Comprensión de datos

En esta etapa se debe hacer una exploración de los datos, en donde se analizará la estructura, la calidad y las características de los datos que se recopilaron en la sección anterior.

Información sobre los campos disponibles útiles a nivel de video:

- VideoID: ID asociado con el video.
- Veracity: Indica si la declaración en el video es una verdad o una mentira: el valor de 0 indica una mentira, el valor de 1 indica una verdad.
- VidLength_sec: Indica la duración del video en segundos.

Información sobre los campos disponibles útiles a nivel de participante:

- TargetID: ID asociado con el participante.
- Sex: Indica el sexo del participante: el valor de 0 indica un participante femenino, el valor de 1 indica un participante masculino.
- Age: Indica la edad del participante en años.
- SchoolYear: Indica el año del participante en la universidad.

En la tabla 1 se puede visualizar una tabla de frecuencias con los campos “sex” y “Veracity”. Se puede ver que se tienen exactamente 160 videos de mentiras y 160 de verdad, y lo mismo se puede apreciar para la cantidad hombres y mujeres.

Sexo \ Veracidad	Veracidad		Total
	Mentira	Verdad	
Femenino	80	80	160
Masculino	80	80	160
Total	160	160	320

Tabla 1: Tabla de frecuencia de las variables “Sex”(sexo) y “Veracity”(veracidad).

En la tabla 2 se puede ver una tabla resumen de la información de los videos.

	Duración (seg.)	Edad
count	320.00	80.00
mean	35.73	20.20
std	3.49	1.55
min	24.33	18.00
25 %	33.93	19.00
50 %	36.07	20.00
75 %	38.21	21.00
max	43.60	26.00

Tabla 2: Resumen de datos de los videos

3.6. Etapa 6: Preparación de datos

En esta etapa se deben preparar los datos que se utilizarán para entrenar a los modelos de aprendizaje profundo (deep learning), en donde el objetivo principal es asegurar que estos datos sean de alta calidad, estén en el formato adecuado, contengan información relevante y sean suficientes para la complejidad del problema.

En primer lugar, la librería librosa de Python trabaja con audios en formato “wav”, por lo tanto los videos pasarán por un proceso de transformación con la ayuda de FFmpeg, una herramienta de línea de comandos ampliamente utilizada para manipular y convertir archivos multimedia.

Para automatizar el proceso de conversión de formato de cada video, se diseñó y programó un proceso en Python que permitió la transformación secuencial de cada archivo de video en el conjunto de datos a su equivalente en formato de audio. El proceso en Python consistió en iterar sobre cada video en el conjunto de datos y ejecutar el comando de FFmpeg desde el código. Los argumentos necesarios para ejecutar la herramienta de FFmpeg incluía la ubicación del archivo de video de entrada, la ubicación del archivo de audio de salida y los parámetros de conversión requeridos. En el anexo A se puede ver un extracto de este código.

Para preparar la estructura de los datos que se utilizarán para el entrenamiento de los modelos, se tomó en consideración los siguientes puntos:

- Se tiene una baja cantidad de datos para enfrentar la complejidad de este problema, por lo que se debe utilizar una técnica de aumento de datos (data augmentation) para conseguir una mayor cantidad de datos.
- Se dividirá cada audio pequeños segmentos del mismo tamaño, ya que deben coincidir con el número de neuronas de entrada del modelo y además esto permitirá un

aumento de datos.

- Se crearán dos grandes conjuntos de datos a partir de los audios. En el primero se utilizará la transformada de Fourier de tiempo reducido (STFT) como base, y otro en donde se utilice los Coeficientes Cepstrales de Mel (MFCC) como base.

Se generaron dos archivos “.txt”, uno con datos de STFT y otro con los datos de MFCC. Estos archivos contienen una tabla de datos en donde la primera columna “labels” guarda información de la veracidad del audio (1 si corresponde a una verdad, 0 si corresponde a una mentira), y otra columna “values” con los datos STFT o MFCC según corresponda. En el anexo C se puede encontrar un extracto con mayor detalle del código programado para la generación de estos archivos.

3.7. Etapa 7: Modelado

En esta etapa se busca desarrollar los modelos predictivos que serán entrenados con los datos generados en la etapa anterior. Específicamente, se desarrollaron 3 modelos de Deep Learning: uno de perceptrón multicapa (MLP), otro de redes neuronales convolucionales (CNN) y un modelo de redes neuronales recurrentes (RNN). Para llevar a cabo la construcción de estos modelos, se utilizó la librería “Keras” de Python. En el Anexo D se puede ver un extracto más detallado del código programado para los modelos.

A continuación, se detallarán los procedimientos realizados en la implementación de cada modelo de acuerdo al flujo establecido:

1. **Partición de los datos:** Se realizaron dos particiones, una para el conjunto de datos STFT y otra para el conjunto de datos MFCC. Cada partición generó un conjunto de datos destinado al entrenamiento del modelo, así como otro conjunto que se reservó la validación del modelo. La división se llevó a cabo de manera que el conjunto de datos de entrenamiento contuviera el 80 % de las muestras totales, mientras que el conjunto de datos de validación se conformara con el 20 % restante.
2. **Construcción y entrenamiento del modelo:** Para cada modelo considerado (MLP, CNN, RNN), se procedió a su construcción, donde se agregaron las capas más representativas a cada tipo de arquitectura. Posteriormente, se llevó a cabo el entrenamiento de cada modelo utilizando los datos de entrenamiento previamente mencionados. Durante el entrenamiento, se realizaron ajustes iterativos de los parámetros del modelo con el objetivo de minimizar la función de pérdida y mejorar su rendimiento predictivo. Una vez finalizado el proceso de entrenamiento, se guardaron tanto el modelo como las métricas asociadas obtenidas durante el entrenamiento. Es importante destacar que no hay una heurística o regla definida para determinar el número de capas o neuronas en un modelo, por lo tanto, se debe repetir este proceso y probar diferentes topologías, variando gradualmente estos parámetros del modelo.

3. **Evaluación y selección del mejor modelo:** Se procedió a la representación gráfica de las métricas obtenidas para cada modelo, con el propósito de realizar una evaluación comparativa de su rendimiento. Esta visualización permitió identificar el modelo que presentara un desempeño superior en términos de las métricas de evaluación consideradas, siendo este seleccionado como el mejor modelo para su uso en tareas de predicción o clasificación.

Variables	Total
x, y	3437
x_{train}, y_{train}	2749
$x_{validation}, y_{validation}$	688

Tabla 3: Cantidad del conjunto de datos utilizados en los modelos.

La tabla 3 presenta un resumen de la cantidad de datos utilizados, en donde:

- x : Es la cantidad de datos totales del conjunto de datos de entrada de STFT y del MFCC (input). Contiene los segmentos de audio utilizados.
- y : Es la cantidad de datos totales del conjunto de datos de salida de datos STFT y del MFCC (output). Contiene la información que indica si el segmento de audio corresponde a una mentira o a una verdad.

Además:

- x_{train}, y_{train} : Es la cantidad de datos totales del conjunto de datos de entrenamiento de STFT y del MFCC. Corresponde al 80 % de los datos totales.
- $x_{validation}, y_{validation}$: Es la cantidad de datos totales del conjunto de datos de validación de STFT y del MFCC. Corresponde al 20 % de los datos totales.

	Datos	
Modelos	STFT	MFCC
MLP y RNN	(302, 1025)	(302, 13)
CNN	(302, 1025, 1)	(302, 13, 1)

Tabla 4: Dimensión de cada dato y en cada modelo.

La tabla 4 muestra las dimensiones de los datos para cada modelo dependiendo del conjunto de datos, en donde:

- **MLP:** Es el modelo perceptrón multicapa.
- **CNN:** Es el modelo de redes neuronales convolucionales.
- **RNN:** Es el modelo de redes neuronales recurrentes.
- **STFT:** Es la Transformada de Fourier de Tiempo Reducido.
- **MFCC:** Son los Coeficientes Cepstrales en las Frecuencias de Mel.

Las funciones STFT y MFCC toman una lista de datos numéricos que representan un segmento de audio y generan una lista de datos bidimensionales. En el anexo E se puede ver el detalle de los parámetros y el cálculo realizado que explica la dimensión de cada caso.

Para los modelos MLP y RNN, los datos se representan en una dimensión bidimensional. Sin embargo, en el caso de los modelos CNN, se requiere una tercera dimensión adicional, ya que estos modelos trabajan con imágenes y esta dimensión tiene ese canal extra para estos datos.

El modelo MLP consta principalmente de varias capas densas. El modelo comienza con una capa de entrada plana, seguida de tres capas densas, cada una con una función de activación ReLU y un regularizador de peso L2 para evitar el sobreajuste. Después de cada capa densa, se agrega una capa de Dropout para regularizar aún más el modelo y reducir el riesgo de sobreajuste. Finalmente, se agrega una capa de salida con una sola neurona y una función de activación "sigmoid". El modelo se compila con el optimizador Adam y la función de pérdida "binary_crossentropy", y se utiliza la métrica de "accuracy" para evaluar su rendimiento durante el entrenamiento. En el Anexo D se puede encontrar más información respecto al código del modelo programado. A continuación, en la figura 9 se presenta un resumen de la topología del modelo MLP.

```

Model: "sequential"
-----
Layer (type)                Output Shape              Param #
=====
flatten (Flatten)           (None, 3926)              0
dense (Dense)                (None, 512)              2010624
dropout (Dropout)           (None, 512)              0
dense_1 (Dense)              (None, 256)              131328
dropout_1 (Dropout)          (None, 256)              0
dense_2 (Dense)              (None, 64)               16448
dropout_2 (Dropout)          (None, 64)               0
dense_3 (Dense)              (None, 1)                65
=====
Total params: 2,171,465
Trainable params: 2,171,465
Non-trainable params: 0
-----

```

Figura 9: Topología MLP.

El modelo CNN consta principalmente de varias capas convolucionales y dos capas densas. El modelo comienza con capas convolucionales que permiten extraer características de los datos de entrada, seguidas de capas de Max Pooling y Batch Normalization para reducir la dimensionalidad y normalizar los datos. La salida se aplanar y se conecta a una capa densa con una función de activación ReLU y una capa de Dropout para evitar el sobreajuste. Finalmente, se agrega una capa de salida con una neurona y una función de activación "sigmoid" para clasificación binaria. El modelo se compila con el optimizador Adam y la función de pérdida "binary_crossentropy", y se utiliza la métrica de "accuracy" para evaluar su rendimiento durante el entrenamiento. En el Anexo D se puede encontrar más información respecto al código del modelo programado. A continuación, en la figura 10 se presenta un resumen de la topología del modelo CNN.

```

Model: "sequential"
-----
Layer (type)                Output Shape              Param #
=====
conv2d (Conv2D)              (None, 300, 11, 32)      320
max_pooling2d (MaxPooling2D) (None, 150, 6, 32)       0
batch_normalization (BatchN (None, 150, 6, 32)      128
ormalization)
conv2d_1 (Conv2D)            (None, 148, 4, 32)      9248
max_pooling2d_1 (MaxPooling2D) (None, 74, 2, 32)       0
batch_normalization_1 (BatchN (None, 74, 2, 32)      128
ormalization)
conv2d_2 (Conv2D)            (None, 73, 1, 32)      4128
max_pooling2d_2 (MaxPooling2D) (None, 37, 1, 32)       0
batch_normalization_2 (BatchN (None, 37, 1, 32)      128
ormalization)
flatten (Flatten)            (None, 1184)             0
dense (Dense)                (None, 100)              118500
dropout (Dropout)            (None, 100)              0
dense_1 (Dense)              (None, 1)                101
=====
Total params: 132,681
Trainable params: 132,489
Non-trainable params: 192
-----

```

Figura 10: Topología CNN.

El modelo RNN consta principalmente de dos capas LSTM y dos capas densas. El modelo comienza con dos capas LSTM que se utilizan para capturar y procesar la información secuencial de los datos. La primera capa LSTM devuelve secuencias completas, lo que permite a la segunda capa LSTM recibir estas secuencias completas y proporcionar una salida final. A continuación, se agrega una capa densa con una función de activación ReLU y una capa de Dropout para regularizar el modelo y evitar el sobreajuste. Finalmente, se agrega una capa de salida con una neurona y una función de activación "sigmoid" para realizar la clasificación binaria. El modelo se compila con el optimizador Adam y la función de pérdida "binary_crossentropy", y se utiliza la métrica de "accuracy" para evaluar su rendimiento durante el entrenamiento. En el Anexo D se puede encontrar más información respecto al código del modelo programado. A continuación, en la figura 11 se presenta un resumen de la topología del modelo RNN.

```
Model: "sequential"
-----
Layer (type)                Output Shape              Param #
-----
lstm (LSTM)                  (None, 302, 64)          19968
lstm_1 (LSTM)                (None, 64)               33024
dense (Dense)                (None, 64)               4160
dropout (Dropout)            (None, 64)               0
dense_1 (Dense)              (None, 1)                65
-----
Total params: 57,217
Trainable params: 57,217
Non-trainable params: 0
-----
```

Figura 11: Topología RNN.

Posteriormente, los modelos fueron entrenados utilizando el conjunto de datos previamente generado. Se estableció un tamaño de lote ("batch_size") de 32 muestras por iteración y se realizaron un total de 50 iteraciones ("epochs") en el conjunto de datos. Finalmente, se guardaron los modelos y el registro de las métricas de entrenamiento y validación.

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

El objetivo de este capítulo es evaluar y validar la solución propuesta, siguiendo las últimas etapas de la metodología de ciencia de datos. Se seleccionará el modelo con las mejores métricas y se implementará en una aplicación web para que pueda ser utilizado.

4.0.1. Etapa 8: Evaluación

En esta etapa se lleva a cabo la evaluación de los modelos generados, teniendo en cuenta los objetivos del negocio predefinidos.

A continuación, en la tabla 5 se presenta un resumen que muestra los resultados obtenidos de la función de pérdida ("Loss") y la precisión ("Accuracy") de los modelos utilizando los conjuntos de datos STFT y MFCC.

Modelos \ Datos	STFT		MFCC	
	Accuracy	Loss	Accuracy	Loss
MLP	0,000 %	8,861	0,000 %	8,366
CNN	45,000 %	1,879	51,162 %	1,074
RNN	33,333 %	0,942	50,145 %	0,938

Tabla 5: Resultados de las métricas de entrenamiento de cada modelo.

La función de pérdida es una medida utilizada para evaluar qué tan bien se ajusta el modelo a los datos. En general, se busca minimizar este valor, ya que cuanto más bajo sea, mejor será el ajuste del modelo a los datos. Por otro lado, la precisión es una métrica que indica la proporción de muestras clasificadas correctamente, en relación al total de muestras evaluadas. Cuanto más cercana a 100 % sea la precisión, mejor será el rendimiento del modelo.

Las mejores métricas se encuentran en los modelos CNN y RNN entrenados con el conjunto de datos MFCC. La precisión de ambos modelos es muy similar, con una diferencia de 1,01 %. Sin embargo, la diferencia en los valores de pérdida es de 0,14, favoreciendo al modelo RNN. Dado que el modelo RNN tiene un valor en la función de pérdida más bajo, se espera que tenga una mejor capacidad de ajuste a los datos en general.

4.0.2. Etapa 9: Implementación

En esta etapa, se lleva a cabo la implementación del prototipo del detector de mentiras en una aplicación web. El objetivo principal de esta aplicación web es permitir a los usuarios

cargar un archivo de audio, analizarlo y determinar si corresponde a una verdad o una mentira.

La aplicación web ha sido desarrollada utilizando Python con Django, y consta de una página web que cuenta con dos botones y un reproductor de audio. El primer botón permite al usuario cargar un archivo de audio, mientras que el segundo botón se encarga de analizar el audio. Además, se ha incorporado un reproductor para que el usuario pueda escuchar el audio cargado.

Detector de mentiras

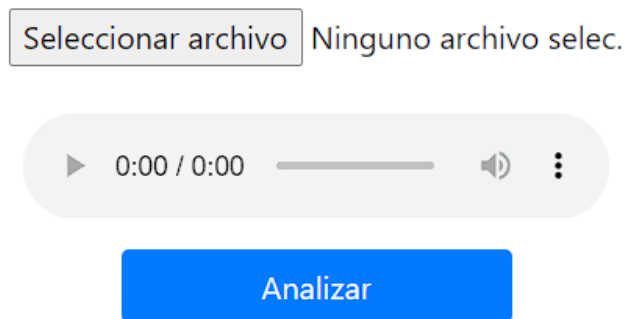


Figura 12: Página principal de la aplicación Web para detectar mentiras.

Al presionar el botón de análisis, el sistema divide el audio en pequeños segmentos, los cuales son analizados y se les asigna una probabilidad de ser verdaderos. Luego, los valores obtenidos de cada segmento se grafican con el objetivo de visualizar la confianza, o grado de veracidad, del audio en función del tiempo.



Figura 13: Ejemplos del gráfico de confianza en función del tiempo en la aplicación web.

4.0.3. Etapa 10: Retroalimentación

En la etapa 10 de mejora continua del modelo, se realizaron pruebas adicionales con el objetivo de mejorar su rendimiento. Se llevaron a cabo dos pruebas específicas: en la primera, se modificó la salida del modelo de 1 neurona a 2 neuronas; y en la segunda, se decidió dividir el audio en dos segmentos en lugar de diez. Estas pruebas generaron un total de cuatro combinaciones posibles, siendo la combinación actual utilizada, aquella con 1 neurona de salida y la división en 10 segmentos.

El propósito de estas pruebas fue evaluar el impacto de estas modificaciones en la precisión y el rendimiento del modelo. Se buscaba determinar si la incorporación de más neuronas en la salida del modelo o una variación en la cantidad de segmentos del audio, podrían proporcionar resultados más precisos y confiables.

A continuación, en la tabla 6 se presenta un resumen de los resultados obtenidos:

Modelos Datos	RNN-1N		RNN-2N	
	Accuracy	Loss	Accuracy	Loss
MFCC-2S	42,603 %	0,775	47,337 %	0,817
MFCC-10S	50,145 %	0,938	50,872 %	0,834

Tabla 6: Resultados de las métricas de entrenamiento de cada modelo.

CAPÍTULO 5

CONCLUSIONES

En este proyecto, se aplicó e integró una metodología para detectar mentiras en audios utilizando modelos de Machine Learning. El trabajo principal se enfocó en el desarrollo de tres aspectos fundamentales; La investigación y análisis de transformadas matemáticas para comprender y representar las señales de onda de los audios; El estudio y comparación de diversos modelos de Machine Learning adecuados para trabajar con audios; Y la evaluación de cada uno de los modelos estudiados para seleccionar el que presentara el mejor rendimiento. Como resultado, se logró el desarrollo de una aplicación web que utiliza Deep Learning para analizar audios y determinar si contienen información falsa o veraz.

La estrategia implementada en este proyecto se dividió en dos fases. En la primera fase, se realizó un estudio de los conceptos técnicos necesarios para abordar el problema de detección de mentiras en audios. En la segunda fase, se aplicaron e integraron estos conceptos en el contexto de la detección de mentiras.

La primera fase se enfocó en el procesamiento y comprensión del sonido. Se exploraron dos transformadas matemáticas ampliamente utilizadas en el estudio del sonido: La Transformada de Fourier de Tiempo Reducido (STFT), y los Coeficientes Ceptrales en las Frecuencias de Mel (MFCC). Estas técnicas permitieron representar y analizar adecuadamente los audios.

Posteriormente, se profundizó en el estudio de modelos de Deep Learning, una subcategoría del Machine Learning, que se centra específicamente en el uso de redes neuronales. Se examinaron tres modelos en particular: el perceptrón multicapa (MLP), las redes neuronales convolucionales (CNN) y las redes neuronales recurrentes (RNN).

Luego, se investigó y adoptó la metodología fundamental de la ciencia de datos, siguiendo un enfoque basado en una guía desarrollada por IBM que consta de diez etapas. Esta metodología proporcionó una estructura sólida y coherente para la ejecución del proyecto, asegurando un proceso ordenado y eficaz en todas las etapas de desarrollo. Por último, se agregó una sección del estado del arte en donde se revisan diversos trabajos sobre la detección de mentiras.

En la segunda fase de aplicación e integración, se inició definiendo la propuesta de solución siguiendo las 10 etapas de la metodología fundamental de la ciencia de datos. Cada etapa desempeñó un papel fundamental en el avance progresivo del proyecto y su posterior ejecución. A continuación, se describen las actividades realizadas en cada una de las 10 etapas:

1. **Comprensión del negocio:** Se identificó y definió el problema, los objetivos del negocio, los criterios de éxito del negocio y los requisitos de solución.
2. **Enfoque analítico:** Se planificaron las actividades principales del proyecto y se estimó

una duración aproximada para cada una. El tiempo estimado total fue de 22 semanas para completar todas las actividades planificadas.

3. **Requisitos de datos:** Se establecieron los requisitos necesarios para los datos a utilizar, especificando que estos debían ser archivos de audio en formato “wav”.
4. **Recopilación de datos:** Se evaluaron dos opciones para obtener los datos: crear un conjunto de datos o buscar uno ya existente. Se decidió utilizar un conjunto de datos previamente catalogados y preparados para su uso.
5. **Comprensión de datos:** Se realizó un análisis exhaustivo y exploratorio de los datos para comprender su estructura, calidad y características inherentes.
6. **Preparación de datos:** Se definieron los pasos necesarios para preparar los datos y cumplir con los requisitos de entrenamiento de los modelos. Esto implicó la transformación de los videos en archivos de audio “wav” y la creación de dos conjuntos de datos distintos utilizando técnicas como STFT y MFCC.
7. **Modelado:** Se delinearon los pasos para la creación y entrenamiento de los modelos. Se realizó una partición de los datos en un 80 % para entrenamiento y un 20 % para validación. Se construyeron tres modelos: MLP, CNN y RNN. Finalmente, se guardaron los modelos entrenados junto con las métricas obtenidas durante el proceso de entrenamiento para su posterior análisis.
8. **Evaluación:** Se evaluaron los modelos utilizando gráficos generados a partir de las métricas obtenidas durante el entrenamiento. Se seleccionó el modelo RNN entrenado con el conjunto de datos MFCC como el más adecuado.
9. **Implementación:** Se desarrolló e implementó un prototipo del detector de mentiras en una aplicación web utilizando Django. Esta aplicación integra el modelo RNN seleccionado y ofrece la funcionalidad de cargar audios para su análisis y predicción de veracidad.
10. **Retroalimentación:** Se implementó un proceso de mejora continua del modelo a lo largo del tiempo, con el objetivo de optimizar su rendimiento. Se realizaron modificaciones adicionales para mejorar los resultados: se ajustó la salida del modelo para tener dos neuronas en lugar de una, y se redujo la división del audio a solo dos segmentos en lugar de diez. Estas modificaciones dieron lugar a tres nuevas variantes del conjunto de datos MFCC, que fueron utilizadas para entrenar el modelo RNN y evaluar su desempeño.

El modelo RNN utilizado logró una precisión de aproximadamente 75 % en el conjunto de datos de entrenamiento y una precisión de aproximadamente 50 % en el conjunto de datos de validación. Además, se observó una función de pérdida de aproximadamente 0,9. La diferencia en la precisión entre los datos de entrenamiento y los datos de validación indica la

presencia de sobreajuste (overfitting). El sobreajuste puede tener varias causas, y en este caso se sospecha que se debe a dos causas principales: la estructura del modelo y una cantidad limitada de datos de entrenamiento.

En relación a la estructura del modelo, puede haber una complejidad excesiva, falta de técnicas de regularización o un conjunto de validación insuficiente para la complejidad del problema. Además, es relevante tener en cuenta que la cantidad de datos recomendada para entrenar un modelo de Deep Learning suele ser de cientos de miles o incluso millones de datos. En este caso particular, se utilizaron aproximadamente 3400 datos en total, lo cual es inferior a lo recomendado para enfrentar la complejidad de este problema.

La limitación de contar con una baja cantidad de datos para abordar el problema puede implicar que exista una alta probabilidad de que estos datos sean atípicos o muy poco representativos del problema que se intenta resolver. Esto puede resultar en un sobreajuste del modelo, donde este aprende más de los datos específicos de entrenamiento que del problema en sí. Como resultado, se observa una precisión del 75 % en el conjunto de datos de entrenamiento, pero solo una precisión del 50 % en el conjunto de datos de validación.

Los principales alcances, limitaciones y puntos de mejora identificados en este proyecto son los siguientes:

■ Alcances:

- Se implementaron técnicas de análisis de audio, como la Transformada de Fourier de Tiempo Reducido (STFT) y los Coeficientes Cepstrales en las Frecuencias de Mel (MFCC), lo que permitió extraer características relevantes del audio para el análisis de detección de mentiras.
- Se desarrollaron tres modelos de Deep Learning (MLP, CNN y RNN).

■ Limitaciones:

- Una limitación importante fue la baja cantidad de datos utilizados en el conjunto de entrenamiento. Para mejorar la generalización del modelo, sería recomendable recopilar una mayor cantidad de datos de audio para entrenar el modelo de manera más efectiva.
- Otra limitación identificada es la utilización exclusiva del audio para el análisis en la detección de mentiras. Dado que los videos recibidos por la entidad incluían tanto voz narrada como diapositivas visuales, no se pudo aprovechar la información visual para mejorar la precisión de la detección. Sería interesante explorar en futuros trabajos el análisis multimodal, que considere tanto el audio como la información visual, para obtener mejores resultados en la detección de mentiras en este contexto específico.

■ Puntos de mejora:

- Aumentar la cantidad de datos de entrenamiento para mejorar la capacidad de generalización del modelo.
- Explorar y utilizar otras técnicas de análisis de audio además de la STFT y los MFCC, como la transformada wavelet, para evaluar su efectividad en la detección de mentiras.
- Considerar la incorporación de información visual de las diapositivas en el análisis, ya sea mediante técnicas de procesamiento de imágenes o el uso de modelos multimodales, con el objetivo de aprovechar al máximo los datos disponibles y mejorar la precisión en la detección de mentiras en el contexto de los videos de presentaciones de proyectos de emprendimiento.

El principal objetivo de este proyecto fue implementar e integrar una metodología para la detección de mentiras en audios utilizando técnicas de Machine Learning. Para lograr este objetivo, se aplicó la metodología fundamental de la ciencia de datos, lo que permitió alcanzar tres objetivos específicos clave que fueron fundamentales para el desarrollo efectivo de este trabajo:

- **Análisis de transformadas matemáticas para la representación de audios:** Se realizaron investigaciones exhaustivas sobre dos técnicas principales para el procesamiento y análisis de audios: la Transformada de Fourier de Tiempo Reducido (STFT) y los Coeficientes Cepstrales en las Frecuencias de Mel (MFCC). Estas transformadas fueron evaluadas y comparadas en términos de su capacidad para extraer características relevantes y discriminativas de los audios utilizados en la detección de mentiras.
- **Estudio y comparación de modelos de Machine Learning:** Se exploraron modelos de Deep Learning, un subcampo del Machine Learning, y se compararon tres modelos específicos: MLP, CNN y RNN. Estos modelos fueron seleccionados debido a su capacidad para aprender patrones complejos y representaciones de alto nivel a partir de los datos de audio.
- **Evaluación de los modelos propuestos y desarrollo de una aplicación web para su uso:** Se evaluaron y compararon los modelos propuestos utilizando diversas métricas de rendimiento. Se realizaron gráficas y análisis detallados para seleccionar el modelo con el mejor rendimiento en la detección de mentiras. Además, se desarrolló una aplicación web que permite utilizar el modelo seleccionado de manera práctica y accesible.

En conclusión, se lograron y cumplieron los objetivos propuestos en este proyecto de detección de mentiras en audios mediante el uso de técnicas de Machine Learning. Se aplicó la metodología fundamental de la ciencia de datos, permitiendo el análisis de transformadas matemáticas para la representación de los audios, el estudio y comparación de modelos de Deep Learning, así como la evaluación de los modelos propuestos y la construcción de una aplicación web para su uso práctico.

Finalmente, como recomendaciones y posibles direcciones futuras, el problema de la detección de mentiras con modelos de Deep Learning presenta una complejidad que requiere una cantidad considerable de datos, en el orden de decenas de miles e incluso cientos de miles de muestras de audio, para abordar y entrenar el modelo de manera efectiva. Además, es importante considerar el uso de transformadas matemáticas que permitan extraer características relevantes del audio. En este proyecto, se utilizó la transformada de Coeficientes Cepstrales en las Frecuencias de Mel (MFCC), que ha demostrado ser eficaz en el procesamiento de señales de audio. Sin embargo, también se sugiere considerar el uso de la transformada wavelet u otras técnicas de extracción de características para evaluar su efectividad en la detección de mentiras.

También se recomienda explorar técnicas de aumento de datos, como la segmentación de los audios en pequeños segmentos o la aplicación de variaciones en la velocidad de reproducción. Estas técnicas pueden enriquecer la diversidad y cantidad de datos disponibles para el entrenamiento, lo que a su vez mejora la capacidad de generalización del modelo.

Además, se sugiere realizar variaciones en la arquitectura del modelo principal. Por ejemplo, podría considerarse la incorporación de capas densas con regularizadores o capas de dropout para prevenir el sobreajuste y aumentar la robustez del modelo.

Por otro lado, en este trabajo se ha desarrollado únicamente un modelo que analiza audios. Sin embargo, también podría ser recomendable explorar enfoques multimodales que permitan la incorporación de diversos tipos de datos, como texto, imagen, audio, video, entre otros. El objetivo principal de un modelo multimodal sería capturar las relaciones e interacciones entre estas modalidades para mejorar el rendimiento en la tarea específica de detección de mentiras.

Estas recomendaciones y posibles direcciones futuras pueden contribuir a mejorar el modelo actual y explorar nuevas formas de abordar el desafío de la detección de mentiras utilizando modelos de Deep Learning.

ANEXOS

Anexo A

A continuación se muestra un extracto del código programado para transformar el conjunto de videos en format “wmv” a un formato de audio “wav”.

Librerías, módulos y variables de configuración:

```
# *** Modules ***
# 1) Python Modules
import os
import time
import subprocess as sp
from pathlib import Path

# 2) Installed Modules
# *** Modules ***

# *** Global variables ***
import config as cfg
BASE_DIR = Path(__file__).parent.parent / 'data'
DATASET_PATH = BASE_DIR / 'C01_Videos'
DES_PATH = BASE_DIR / 'C02_audios'
ffmpeg = cfg.ffmpeg
FFMPEG = Path(ffmpeg)
# *** Global variables ***
```

Funciones:

```

# *** Functions ***
def get_audio(file_path, des_path_wav):
    command = f"{'FFMPEG'} -i {file_path} {des_path_wav}"
    print(f"Ejecutando: {command}")
    sp.call(command, shell=True)

def save_audios():
    for dirpath, dirnames, filenames in os.walk(DATASET_PATH):
        total_files = len(filenames)
        print(f"dirpath: {dirpath}")
        print(f"dirnames: {dirnames}")
        print(f"len(filenames): {total_files}\n")
        i = 0
        for file_name in filenames:
            i += 1
            # file_path = os.path.join(dirpath, file_name)
            file_path = Path(dirpath) / file_name
            des_path_wav = DES_PATH / (file_name.split('.')[0] \
                + '.wav')
            print(f"*** ***** Ejecutando {i}/{total_files} " \
                "***** **")
            get_audio(file_path, des_path_wav)
            print(f"*** Archivo procesado correctamente: " \
                "{file_name} ***\n")
# *** Functions ***

```

Función principal:

```

# *** Main ***
if __name__ == '__main__':
    start = time.time()
    save_audios()
    end = time.time()
    print(f"Total time: {round(end - start, 3)} seconds")
# *** Main ***

```

Anexo B

A continuación se muestra un extracto del código programado para generar los gráficos de las figuras ??, ?? y ??.

Librerías, módulos y variables de configuración:

```
# *** Modules ***
# 1) Python Modules
import time
from pathlib import Path

# 2) Installed Modules
import numpy as np
import matplotlib.pyplot as plt
import librosa, librosa.display
# *** Modules ***

# *** Global variables ***
import config as cfg
BASE_DIR = Path(__file__).parent
audio_name = cfg.audio_name
TOTAL_PATH = BASE_DIR.parent / 'data' / 'C02_audios' / audio_name

FIG_SIZE = (15, 10)
# *** Global variables ***
```

Función que grafica la amplitud de onda vs tiempo.

```
def get_waveform(sr=44100):
    signal, sample_rate = librosa.load(TOTAL_PATH, sr=sr)

    # Plot waveform
    plt.figure(figsize=FIG_SIZE)
    librosa.display.waveshow(signal, sr=sample_rate)
    plt.xlabel("Tiempo (s)")
    plt.ylabel("Amplitud")
    plt.title("Forma de la onda")
    plt.show()
```

Función que grafica y calcula la Transformada de Fourier de Tiempo Reducido - STFT (espectrograma).

```
def get_stft(sr=22050, n_fft=2048, hop_length=512):  
    signal, sample_rate = librosa.load(TOTAL_PATH, sr=sr)  
  
    # stft  
    stft = librosa.stft(signal, n_fft=n_fft, hop_length=hop_length)  
  
    # calculate abs values on complex numbers to get magnitude  
    spectrogram = np.abs(stft)  
  
    # apply logarithm to cast amplitude to Decibels  
    log_spectrogram = librosa.amplitude_to_db(spectrogram)  
  
    # display spectrogram  
    librosa.display.specshow(  
        log_spectrogram,  
        sr=sample_rate,  
        hop_length=hop_length  
    )  
    plt.xlabel("Tiempo")  
    plt.ylabel("Frecuencia")  
    plt.colorbar(format="+2.0f dB")  
    plt.title("Espectrograma (dB)")  
    plt.show()
```

Función que gráfica y calcula los Coeficientes Cepstrales de Mel (MFCC).

```
def get_mfcc(sr=22050, n_fft=2048, hop_length=512, n_mfcc=13):  
    signal, sample_rate = librosa.load(TEST_PATH, sr=sr)  
  
    # MFCCs  
    MFCCs = librosa.feature.mfcc(  
        signal,  
        sample_rate,  
        n_fft=n_fft,  
        hop_length=hop_length,  
        n_mfcc=n_mfcc  
    )  
  
    # display MFCCs  
    plt.figure(figsize=FIG_SIZE)  
    librosa.display.specshow(  
        MFCCs,  
        sr=sample_rate,  
        hop_length=hop_length  
    )  
    plt.xlabel("Tiempo")  
    plt.ylabel("Coeficientes MFCC")  
    plt.colorbar()  
    plt.title("MFCCs")  
    plt.show()
```

Anexo C

Generación de archivo JSON para STFT. Librerías, módulos y variables de configuración:

```
# *** Modules ***
# 1) Python Modules
import os
import time
import math
import json
from pathlib import Path

# 2) Installed Modules
import librosa, librosa.display
import numpy as np
import pandas as pd
# *** Modules ***

# *** Global variables ***
BASE_DIR = Path(__file__).parent.parent / 'data'
DATASET_PATH = BASE_DIR / 'C02_audios'
DES_PATH = BASE_DIR / 'C03_dataset'

SAMPLE_RATE = 44100
TRACK_DURATION = 35
SAMPLES_PER_TRACK = SAMPLE_RATE * TRACK_DURATION
NUM_SEGMENTS = 10
DATA_PATH = DES_PATH / f'stft_data_{NUM_SEGMENTS}.json'

data = {
    "labels": [],
    "stft": [],
}
# *** Global variables ***
```

Función que prepara las rutas de cada archivo y guarda los datos:

```
def save_stft(n_fft=2048, hop_length=512):
    samples_per_segment = int(SAMPLES_PER_TRACK / NUM_SEGMENTS)
    num_vectors_per_segment = math.ceil(samples_per_segment /
                                         hop_length)

    for dirpath, dirnames, filenames in os.walk(DATASET_PATH):
        i = 0
        j = 0
        for file_name in filenames:
            file_path = Path(dirpath) / file_name
            file_name_total = file_name.split('.')[0]
            i += 1

            # *** STFT ***
            if i % 10 == 0:
                j += 1
                file_name_json = f"stft_data_{NUM_SEGMENTS}_n{j}.json"
                json_path_j = DES_PATH / file_name_json
                with open(json_path_j, "w") as des_file:
                    json.dump(data, des_file, indent=4)
                data['labels'] = []
                data['stft'] = []
            # *** STFT ***

        get_stft(file_name_total, file_path, n_fft, hop_length,
                 samples_per_segment, num_vectors_per_segment)
```

Función que genera los datos STFT:

```
def get_stft(file_name_total, file_path, n_fft, hop_length,
             samples_per_segment, num_vectors_per_segment):
    # True or Lie
    veracity = file_name_total[-1:]
    veracity_bool = 1 if veracity == 'T' else 0

    # Load audio file
    signal, sample_rate = librosa.load(file_path, sr=SAMPLE_RATE)
    num_segments_real = math.ceil(len(signal)/samples_per_segment)
    df_total = pd.DataFrame()
    for d in range(num_segments_real):
        start = samples_per_segment * d
        finish = start + samples_per_segment

        # extract stft
        stft = librosa.stft(y=signal[start:finish], n_fft=n_fft,
                           hop_length=hop_length)
        spectrogram = np.abs(stft)
        log_spectrogram = librosa.amplitude_to_db(spectrogram)
        log_spectrogram = log_spectrogram.T

        if len(log_spectrogram) < num_vectors_per_segment:
            diff_shape = np.subtract((num_vectors_per_segment,
                                      int(n_fft/2) + 1), log_spectrogram.shape)
            log_spectrogram = np.pad(log_spectrogram, ((0,
                                                         diff_shape[0]), (0, diff_shape[1])), mode='constant')

        data["labels"].append(veracity_bool)
        data["stft"].append(log_spectrogram.tolist())
```

Generación de archivo JSON para MFCC. Librerías, módulos y variables de configuración:

```
# *** Modules ***
# 1) Python Modules
import os
import time
import math
import json
from pathlib import Path

# 2) Installed Modules
import numpy as np
import pandas as pd
import librosa, librosa.display
# *** Modules ***

# *** Global variables ***
BASE_DIR = Path(__file__).parent.parent / 'data'
DATASET_PATH = BASE_DIR / 'C02_audios'
DES_PATH = BASE_DIR / 'C03_dataset' / 'mfcc'
SAMPLE_RATE = 44100
TRACK_DURATION = 35
SAMPLES_PER_TRACK = SAMPLE_RATE * TRACK_DURATION
NUM_SEGMENTS = 10 # 10, 2
DATA_PATH = DES_PATH / f'mfcc_data_{NUM_SEGMENTS}.json'

data = {
    "labels": [],
    "mfcc": [],
}
# *** Global variables ***
```

Función que prepara las rutas de cada archivo y guarda los datos:

```
def save_mfccs(num_mfcc=13, n_fft=2048, hop_length=512):
    samples_per_segment = int(SAMPLES_PER_TRACK / NUM_SEGMENTS)
    num_mfcc_vectors_per_segment = math.ceil(
        samples_per_segment / hop_length)

    for dirpath, dirnames, filenames in os.walk(DATASET_PATH):
        i = 0
        for file_name in filenames:
            file_path = Path(dirpath) / file_name
            file_name_total = file_name.split('.')[0]
            i += 1
            get_mfcc(
                file_name_total,
                file_path,
                num_mfcc,
                n_fft,
                hop_length,
                samples_per_segment,
                num_mfcc_vectors_per_segment
            )

    # save MFCCs to json file
    with open(JSON_PATH, "w") as des_file:
        json.dump(data, des_file, indent=4)
```

Función que genera los datos MFCC:

```
def get_mfcc(file_name_total, file_path, num_mfcc, n_fft,
             hop_length, samples_per_segment, num_mfcc_vectors_per_segment):
    veracity = file_name_total[-1:]
    veracity_bool = 1 if veracity == 'T' else 0
    signal, sample_rate = librosa.load(file_path, sr=SAMPLE_RATE)
    num_segments_real = math.ceil(len(signal)/samples_per_segment)

    df_total = pd.DataFrame()
    for d in range(num_segments_real):
        start = samples_per_segment * d
        finish = start + samples_per_segment
        mfcc = librosa.feature.mfcc(y=signal[start:finish],
                                   sr=sample_rate, n_mfcc=num_mfcc, n_fft=n_fft,
                                   hop_length=hop_length)
        mfcc = mfcc.T
        if len(mfcc) < num_mfcc_vectors_per_segment:
            diff_shape = np.subtract((num_mfcc_vectors_per_segment,
                                      num_mfcc), mfcc.shape)
            mfcc = np.pad(mfcc, ((0, diff_shape[0]),
                                (0, diff_shape[1])), mode='constant')

        data["labels"].append(veracity_bool)
        data["mfcc"].append(mfcc.tolist())
```

Anexo D

Modelo MLP:

```
def build_model(input_shape):  
    # build network topology  
    model = keras.Sequential()  
  
    # input layer  
    model.add(keras.layers.Flatten(input_shape=input_shape))  
  
    # 1st dense layer  
    model.add(keras.layers.Dense(512, activation='relu',  
                                  kernel_regularizer=keras.regularizers.l2(0.001)))  
    model.add(keras.layers.Dropout(0.3))  
  
    # 2nd dense layer  
    model.add(keras.layers.Dense(256, activation='relu',  
                                  kernel_regularizer=keras.regularizers.l2(0.001)))  
    model.add(keras.layers.Dropout(0.3))  
  
    # 3rd dense layer  
    model.add(keras.layers.Dense(64, activation='relu',  
                                  kernel_regularizer=keras.regularizers.l2(0.001)))  
    model.add(keras.layers.Dropout(0.3))  
  
    # output layer  
    keras.layers.Dense(1, activation='sigmoid')  
  
    # compile model  
    optimiser = keras.optimizers.Adam(learning_rate=0.0001)  
    model.compile(  
        optimizer=optimiser,  
        loss='binary_crossentropy',  
        metrics=['accuracy']  
    )  
  
    return model
```

Modelo CNN:

```
def build_model(input_shape):  
    # build network topology  
    model = keras.Sequential()  
  
    # 1st conv layer  
    model.add(keras.layers.Conv2D(32, (3, 3), activation='relu',  
        input_shape=input_shape))  
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2),  
        padding='same'))  
    model.add(keras.layers.BatchNormalization())  
  
    # 2nd conv layer  
    model.add(keras.layers.Conv2D(32, (3, 3), activation='relu'))  
    model.add(keras.layers.MaxPooling2D((3, 3), strides=(2, 2),  
        padding='same'))  
    model.add(keras.layers.BatchNormalization())  
  
    # 3rd conv layer  
    model.add(keras.layers.Conv2D(32, (2, 2), activation='relu'))  
    model.add(keras.layers.MaxPooling2D((2, 2), strides=(2, 2),  
        padding='same'))  
    model.add(keras.layers.BatchNormalization())  
  
    # flatten output and feed it into dense layer  
    model.add(keras.layers.Flatten())  
    model.add(keras.layers.Dense(100, activation='relu'))  
    model.add(keras.layers.Dropout(0.5))  
  
    # output layer  
    model.add(keras.layers.Dense(1, activation='sigmoid'))  
  
    # compile model  
    optimiser = keras.optimizers.Adam(learning_rate=0.0001)  
    model.compile(  
        optimizer=optimiser,  
        loss='binary_crossentropy',  
        metrics=['accuracy']  
    )  
  
    return model
```

Modelo RNN:

```
def build_model(input_shape):  
    # build network topology  
    model = keras.Sequential()  
  
    # 2 LSTM layers  
    model.add(keras.layers.LSTM(64, input_shape=input_shape,  
                                return_sequences=True))  
    model.add(keras.layers.LSTM(64))  
  
    # dense layer  
    model.add(keras.layers.Dense(64, activation='relu'))  
    model.add(keras.layers.Dropout(0.5))  
  
    # output layer  
    model.add(keras.layers.Dense(1, activation='sigmoid'))  
  
    # compile model  
    optimiser = keras.optimizers.Adam(learning_rate=0.0001)  
    model.compile(  
        optimizer=optimiser,  
        loss='binary_crossentropy',  
        metrics=['accuracy']  
    )  
  
    return model
```

Entrenar y guardar el modelo junto a sus métricas:

```
# train model  
history = model.fit(X_train, y_train,  
                    validation_data=(X_validation, y_validation),  
                    batch_size=32, epochs=NUMBER_EPOCHS)  
  
# save model  
model.save(MODEL_FILE_PATH)  
  
# save history  
with open(HISTORY_FILE_PATH, 'wb') as file:  
    pickle.dump(history.history, file)
```

Anexo E

Parámetros:

- **SAMPLE_RATE** = 44100 [H]: Frecuencia de muestreo.
- **MEAN_TRACK_DURATION** = 35 [s]: Aproximación del promedio de la duración de los audios.
- **SAMPLES_PER_TRACK** = 1543,5 [kH]: Cantidad promedio de muestras por audio.
- **NUM_FFT** = 2048 [H]: Intervalo de muestras consideraras para aplicar la transformada de Fourier (FFT).
- **HOP_LENGTH** = 512 [H]: Ventana de salto de Fourier.
- **NUM_TOTAL_PER_SEGMENT** = 302: Número total de datos por segmento de audio aproximado hacia arriba. Se calcula dividiendo **SAMPLES_PER_TRACK** por **HOP_LENGTH**.
- **NUM_MFCC** = 13: Número total de coeficientes cepstrales de Mel utilizados.

Considerando que $(n \times m)$ es la dimensión de STFT, entonces:

- n : Se calcula en base a la siguiente ecuación

$$n = \frac{\text{NUM_FFT}}{2} + 1 = 1025 \quad (26)$$

- $m = 302$: Corresponde a la variable **NUM_TOTAL_PER_SEGMENT** definida anteriormente.
- Aplicando la transpuesta de la matriz, se obtiene que:

$$(n \times m)^T = (m \times n) = (302 \times 1025) \quad (27)$$

Considerando que $(n \times m)$ es la dimensión de MFCC, entonces:

- $n = 13$: Corresponde a la variable **NUM_MFCC** definida anteriormente.
- $m = 302$: Corresponde a la variable **NUM_TOTAL_PER_SEGMENT** definida anteriormente.
- Aplicando la transpuesta de la matriz, se obtiene que:

$$(n \times m)^T = (m \times n) = (302 \times 13) \quad (28)$$

Anexo F

Contenido adicional del capítulo 2, Marco Conceptual, sobre el procesamiento del sonido.

F.1. Procesamiento del sonido

F.1.1. Sonido

El sonido se define como una pequeña alteración de la presión atmosférica producida por la oscilación de partículas de un medio (fluido o sólido), por las cuales se transmiten ondas mecánicas [Jaramillo, 2007]. En palabras más sencillas, el sonido es la propagación de ondas a través de un medio.

Definiciones importantes a considerar:

- **Ruido:** Son aquellos sonidos no deseados.
- **Período (T):** Es el tiempo en segundos de un ciclo completo de la onda. Es el inverso de la frecuencia.
- **Frecuencia (f):** Es el número de ciclos u oscilaciones que se repiten en un segundo. Es el inverso del período. El sonido audible por las personas se encuentra entre 20 y 20000 hercios y se les denomina ondas sonoras.
- **Amplitud (A):** Es la presión sonora o fuerza por unidad de superficie de las partículas del medio en un punto dado (la onda sonora altera mínimamente la presión atmosférica). La amplitud de onda sonora se puede medir en Pascales (Pa) o en decibeles (Db).

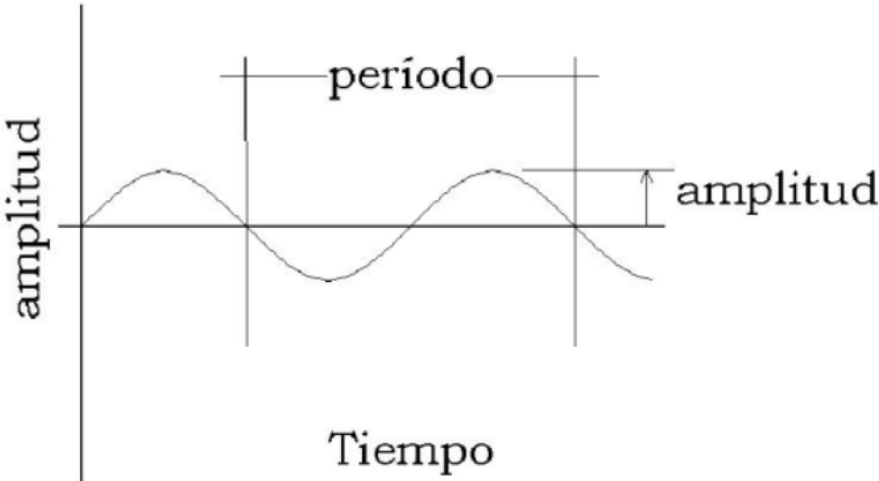


Figura 14: Características de una onda sinusoidal [Jaramillo, 2007].

Una onda se puede modelar como senoide o cosenoide, la cual se puede definir matemáticamente de la siguiente forma [Camastra y Vinciarelli, 2015]:

$$s(t) = A \sin(2\pi ft + \phi) = A \sin\left(\frac{2\pi}{T}t + \phi\right) \quad (29)$$

Donde A es la amplitud y representa la máxima distancia desde la posición de equilibrio, T es el periodo y es el tamaño del intervalo de tiempo entre dos instantes donde $s(t)$ toma el mismo valor, f es la frecuencia y mide el número de veces que $s(t)$ completa un ciclo por segundo, y por último ϕ es la fase y representa el movimiento de la onda.

F.1.2. Conversión Analógica-Digital

La Conversión Analógica-Digital (ADC) es un sistema que transcribe señales analógicas en señales digitales, como por ejemplo las ondas sonoras que capta un micrófono [Velazquez *et al.*, 1998]. La ADC está compuesta de 3 procesos esenciales para su funcionamiento [?]:

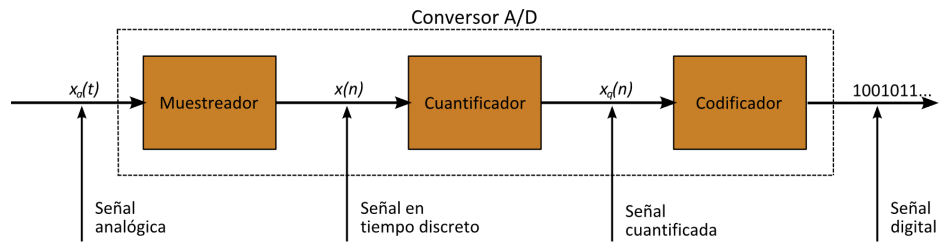


Figura 15: Esquema de la ADC [Commons, 2021].

1. **Muestreo:** Es la cantidad de veces que se mide el valor (Amplitud) de una señal continua en un periodo de tiempo.
2. **Cuantificación:** Es la discretización de los valores obtenidos en el proceso de muestreo. La calidad de esta aproximación dependerá de la cantidad de bits que se utilicen por muestra.
3. **Codificador:** Es la traducción de los valores de tensión eléctrica analógicos obtenidos en el proceso de cuantificación, al sistema binario.

En la imagen 16 se puede ver como a una señal continua, representada por una línea azul, se le aplica un muestreo de datos (representados por los puntos rojos), y por último se le aplica una cuantificación (representada por las líneas verdes).



Figura 16: Cuantificación de una señal de audio [Commons, 2020].

Anexo G

Contenido adicional del capítulo 2, Marco Conceptual, sobre las redes neuronales artificiales.

G.1. Redes Neuronales Artificiales (ANN)

Las Redes Neuronales Artificiales, o ANN (Artificial Neural Network), son un conjunto de algoritmos creados con el objetivo de emular el funcionamiento sistema nervioso del cerebro humano. El sistema nervioso está compuesto por un conjunto de células llamadas neuronas que se encargan de transmitir señales eléctricas para comunicar el cerebro con el cuerpo humano. En 1943 las personas McCulloch y Pitts modelaron una neurona artificial como una neurona que recibe y envía información a otras neuronas. Estas neuronas pueden ser activadas o permanecer inactivas, dependiendo de la información que reciban. [McCulloch y Pitts, 1943]

Una neurona se compone principalmente por tres elementos: Los Inputs, los pesos w_i y una función de activación. En la figura 17 se puede visualizar los elementos de una neurona mencionados anteriormente. [Nielsen, 2015b].

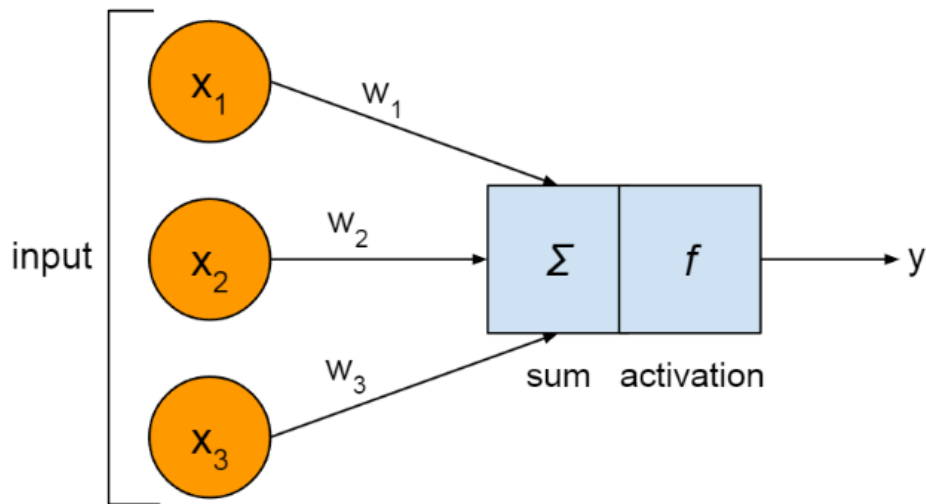


Figura 17: Esquema de una Neurona [Velardo, 2020].

Los componentes de la neurona se pueden definir de la siguiente manera:

- **Inputs:** Son datos de entrada x_i de la neurona.
- **Pesos:** Son los valores que multiplican a los datos de entrada y definen cuanto “peso” tiene cada conexión en la neurona. Luego, los resultados obtenidos de cada conexión se suman, es decir:

$$h = \sum_i x_i \cdot w_i \quad (30)$$

- **Función de activación:** Es una función que toma los datos de entrada, ya procesados

con los pesos, y genera el resultado y , es decir:

$$y = f(h) = f\left(\sum_i x_i \cdot w_i\right) \quad (31)$$

Una ANN se compone principalmente de capas de neuronas, en donde se puede encontrar una capa de entrada, luego una o varias capas ocultas y por último, una capa de salida. La capa de entrada (Input Layer) es la primera capa de la red neuronal, ésta recibe los datos o señales del entorno y no se produce procesamiento de los datos. La capa oculta (Hidden Layer) recibe datos de la capa de entrada o de otras capas ocultas, los cuales son procesados con diversas funciones matemáticas. Por último, está la capa de salida (Output Layer), la cual recibe los datos de la capa oculta y genera el resultado final de toda la red neuronal [Krogh, 2008]. En la figura 18 se puede visualizar un ejemplo de una red neuronal artificial interconectada, la cual posee una capa de entrada, una capa oculta y una capa de salida.

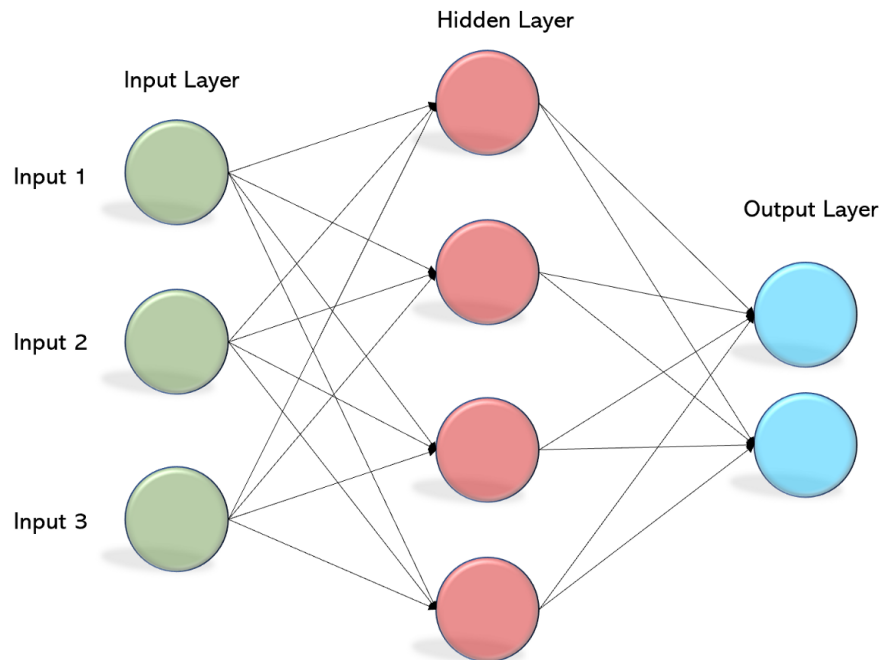


Figura 18: Red Neuronal Artificial [Mohanty, 2019].

Anexo H

Contenido adicional del capítulo 2, Marco Conceptual, sobre tipos de aprendizaje y los entrenamientos de los ANN.

H.1. Aprendizaje y entrenamiento

La fase de aprendizaje y entrenamiento de una red neuronal consiste principalmente en actualizar los pesos de las conexiones de las neuronas de las distintas capas de la red, de tal forma que le permita desempeñar su tarea correctamente. El aprendizaje se puede realizar de tres maneras diferentes [Lanzarini, 2004]:

- **Aprendizaje supervisado:** Se proporciona un conjunto de datos de entrada y de salida para el entrenamiento del modelo. Estos datos se utilizan para actualizar los pesos de la red, de tal forma que se pueda minimizar el error de la diferencia entre los datos de salida generados con los datos de salida proporcionados. Comúnmente se transforma en un problema de optimización, por lo tanto, se pueden utilizar tanto métodos de búsqueda global como métodos de búsqueda local.
- **Aprendizaje no supervisado:** Se proporciona un conjunto de datos de entrada para el entrenamiento del modelo. Estos datos se utilizan para actualizar los pesos de la red, según la correlación que exista entre los datos de entrada.
- **Aprendizaje por refuerzo:** Se proporciona un conjunto de datos de entrada para el entrenamiento del modelo. El modelo utiliza los datos de entrada para generar los datos de salida, luego le se indica si el resultado obtenido es correcto o no para que pueda actualizar los pesos de la red.

Se pueden calcular los resultados de capa de una red neuronal de la siguiente forma:

- **Input Layer:** Es la capa en donde se procesan los datos de entrada de la red. En esta capa no se realiza un procesamiento matemático en las neuronas, por lo tanto, el vector de datos de entrada es igual a los datos de salida, es decir:

$$\begin{aligned} X &= H^{(1)} = A^{(1)} \\ X &= [x_1, x_2, \dots, x_n] \\ H^{(1)} &= [h_1^{(1)}, h_2^{(1)}, \dots, h_n^{(1)}] \\ A^{(1)} &= [a_1^{(1)}, a_2^{(1)}, \dots, a_n^{(1)}] \end{aligned} \tag{32}$$

Donde:

- $x_i, h_i^{(1)}, a_i^{(1)}$: Es el valor de la neurona i .
 - $X, H^{(1)}, A^{(1)}$: Es un conjunto de valor de $x_i, h_i^{(1)}, a_i^{(1)}$ respectivamente.
 - n : Es el número de neuronas de entrada.
- **Hidden Layer:** Es la capa que procesa la información que recibe de otras capas. Puede recibir datos de la capa de entrada y de otras capas ocultas. En esta capa, cada neurona

recibe los datos generados por otras neuronas conectadas a esta, luego estos datos se multiplican por el peso de esa conexión para posteriormente ser sumados y evaluados en la función de activación de la neurona, es decir:

$$H^{(j)} = A^{j-1} \cdot W^{j-1} = \sum_i a_i^{(j-1)} \cdot w_i^{(j-1)} \quad (33)$$

$$A^{(j)} = f(H^{(j)}) \quad (34)$$

Donde:

- A^{j-1} : Conjunto de datos que envían las neuronas de la capa $j - 1$.
 - W^{j-1} : Conjunto de datos de los pesos de la conexión de las neuronas de la capa $j - 1$ con la capa j .
 - $H^{(j)}$: Conjunto de datos de entrada de las neuronas de la capa j .
 - $A^{(j)}$: Conjunto de datos de salida de las neuronas de la capa j .
 - $f(\dots)$: Función de activación.
- **Output Layer:** En la capa que produce los datos de salida de todo el modelo. El proceso de cálculo es similar al anterior mencionado:

$$H^{(m)} = A^{m-1} \cdot W^{m-1} = \sum_i a_i^{(m-1)} \cdot w_i^{(m-1)} \quad (35)$$

$$A^{(m)} = f(H^{(m)}) \quad (36)$$

Donde:

- A^{m-1} : Conjunto de datos que envían las neuronas de la capa $m - 1$.
- W^{m-1} : Conjunto de datos de los pesos de la conexión de las neuronas de la capa $m - 1$ con la capa m .
- $H^{(m)}$: Conjunto de datos de entrada de las neuronas de la capa m .
- $A^{(m)}$: Conjunto de datos de salida de las neuronas de la capa m .
- $f(\dots)$: Función de activación.
- m : Es la posición de la última capa (capa de salida).

REFERENCIAS BIBLIOGRÁFICAS

- [Advice, 2020] Advice, C. (2020). Real estate and investment scams. <https://consumer.ftc.gov/articles/real-estate-investment-scams>. [Online; consultado el 14-Julio-2022].
- [Aizawa et al., 2004] Aizawa, K., Nakamura, Y., y Satoh, S. (2004). *Advances in Multimedia Information Processing-PCM 2004: 5th Pacific Rim Conference on Multimedia, Tokyo, Japan, November 30-December 3, 2004, Proceedings, Part II*, volumen 3332. Springer.
- [Albiol et al., 1998] Albiol, A., Domínguez, L. V., y Prades, J. (1998). *Tratamiento digital de la señal*. Universidad Politécnica de Valencia, Servicio de Publicaciones, 1 edición.
- [Anta, 2012] Anta, J. Á. (2012). Detección del engaño: polígrafo vs análisis verbo-corporal. *Quadernos de criminología: revista de criminología y ciencias forenses*, (19):36–46.
- [Becerra et al., 1989] Becerra, A., Sánchez, F., y Carrera, P. (1989). Indicadores aislados versus patrón general expresivo en la detección de la mentira. *Estudios de psicología*, 10(38):21–29.
- [Camastra y Vinciarelli, 2015] Camastra, F. y Vinciarelli, A. (2015). *Machine learning for audio, image and video analysis: theory and applications*. Springer.
- [Chabria, 2019] Chabria, A. (2019). Fueron condenados injustamente y liberados de prisión pero los siguen tratando como culpables. <https://www.latimes.com/espanol/eeuu>. [Online; consultado el 14-Julio-2022].
- [Commons, 2020] Commons, W. (2020). Señal cuantificada. https://upload.wikimedia.org/wikipedia/commons/a/a8/Senal_cuantificado.png. [Online; consultado el 17-Julio-2022].
- [Commons, 2021] Commons, W. (2021). Conversor ad. https://upload.wikimedia.org/wikipedia/commons/1/14/Conversor_AD.svg. [Online; consultado el 17-Julio-2022].
- [Digevo, 1997] Digevo (1997). Digevo. <https://digevo.com/quienes-somos/>. [Online; consultado el 20-Abril-2022].
- [Dollinger, 2019] Dollinger, A. (2019). Era inocente y pasó 33 años en prisión. <https://www.nytimes.com/es/2019/05/24/espanol/condena-inocente-liberado-prision.html>. [Online; consultado el 14-Julio-2022].
- [Fabesgl, 2015] Fabesgl (2015). Wav vs mp3 – beginners guide. <https://www.off-the-beat.com/wav-vs-mp3/>. [Online; consultado el 10-Marzo-2023].
- [Fernández, 2006] Fernández, R. G. (2006). La mentira. un arte con historia. *Aposta. Revista de Ciencias Sociales*, (26):1–17.

- [Giannakopoulos y Pikrakis, 2014] Giannakopoulos, T. y Pikrakis, A. (2014). *Introduction to audio analysis: a MATLAB® approach*. Academic Press.
- [Gogate et al., 2017] Gogate, M., Adeel, A., y Hussain, A. (2017). Deep learning driven multimodal fusion for automated deception detection. En *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–6. IEEE.
- [Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., y Courville, A. (2016). *Deep learning*. MIT press.
- [Hinojosa, 2020] Hinojosa, S. (2020). Las mentiras de la política. <https://www.lavanguardia.com/politica/20200621/481881640032/verdad-politicos-mentira-lenguaje.html>. [Online; consultado el 14-Julio-2022].
- [Hugenberg et al., 2017] Hugenberg, K., McConnell, A. R., Kunstman, J. W., Lloyd, E. P., Deska, J. C., y Humphrey, B. (2017). Miami university deception detection database.
- [Jaramillo, 2007] Jaramillo, A. M. J. (2007). *Acústica: la ciencia del sonido*. ITM.
- [Krishnamurthy et al., 2018] Krishnamurthy, G., Majumder, N., Poria, S., y Cambria, E. (2018). A deep learning approach for multimodal deception detection. *arXiv preprint arXiv:1803.00344*.
- [Krogh, 2008] Krogh, A. (2008). What are artificial neural networks? *Nature biotechnology*, 26(2):195–197.
- [Lanzarini, 2004] Lanzarini, L. (2004). Redes neuronales artificiales. un enfoque práctico. *Journal of Computer Science & Technology*, 4(2):122–124.
- [LeCun et al., 2015] LeCun, Y., Bengio, Y., y Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.
- [Lee, 2016] Lee, K. (2016). If the pros can't tell when children lie, how can courts decide custody? <https://www.divorceny.com/custody/children-lying/>. [Online; consultado el 17-Noviembre-2021].
- [Martinez, 2021] Martinez, E. (2021). Mentiras blancas, negras y azules, ¿en qué se diferencian? <https://www.psicoactiva.com/blog/mentiras-blancas-negras-y-azules/>. [Online; consultado el 14-Julio-2022].
- [Masip, 2005] Masip, J. (2005). ¿ se pillan antes a un mentiroso que a un cojo? sabiduría popular frente a conocimiento científico sobre la detección no-verbal del engaño. *Papeles del psicólogo*, 26(92):78–91.
- [Masip, 2017] Masip, J. (2017). Deception detection: State of the art and future prospects. *Psicothema*, 29(2):149–159.
- [MathWorks, 2019] MathWorks (2019). stft. <https://www.mathworks.com/help/signal/ref/stft.html>. [Online; consultado el 20-Abril-2023].

- [McCulloch y Pitts, 1943] McCulloch, W. S. y Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.
- [Mohanty, 2019] Mohanty, A. (2019). Multi layer perceptron (mlp) models on real world banking data. https://miro.medium.com/max/1400/1*-IPQ10d46d1sutIbUq1Zcw.png. [Online; consultado el 17-Julio-2022].
- [Nair y Hinton, 2010] Nair, V. y Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. En *lcm1*.
- [Nielsen, 2015a] Nielsen, M. A. (2015a). *Neural networks and deep learning*, volumen 25, capítulo 6. Determination press San Francisco, CA, USA.
- [Nielsen, 2015b] Nielsen, M. A. (2015b). *Neural networks and deep learning*, volumen 25, capítulo 1. Determination press San Francisco, CA, USA.
- [Olah, 2015] Olah, C. (2015). Understanding lstm networks. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. [Online; consultado el 20-Noviembre-2022].
- [Proakis y Manolakis, 1996] Proakis, J. G. y Manolakis, D. G. (1996). *Digital signal processing. Principles, algorithms, and applications*. John G. Proakis and Dimitris G. Manolakis.
- [Provost y Fawcett, 2013] Provost, F. y Fawcett, T. (2013). *Data Science for Business: What you need to know about data mining and data-analytic thinking*. .°Reilly Media, Inc.”.
- [Rollins, 2015] Rollins, J. B. (2015). Metodología fundamental para la ciencia de datos. <https://www.ibm.com/downloads/cas/6RZMKDN8>. [Online; consultado el 20-Noviembre-2022].
- [Serrano, 2019] Serrano, A. G. (2019). Visión artificial con redes convolucionales (cnn). <https://www.ellaberintodefalken.com/2019/10/vision-artificial-redes-convolucionales-CNN.html>. [Online; consultado el 20-Noviembre-2022].
- [Velardo, 2020] Velardo, V. (2020). Deep learning for audio with python. <https://github.com/musikalkemist/DeepLearningForAudioWithPython/blob/master/3-Implementing%20an%20artificial%20neuron%20from%20scratch/slides/3-Implementing%20an%20artificial%20neuron%20from%20scratch.pdf>. [Online; consultado el 17-Julio-2022].
- [Velazquez et al., 1998] Velazquez, S. R., Nguyen, T. Q., y Broadstone, S. R. (1998). Design of hybrid filter banks for analog/digital conversion. *IEEE transactions on signal processing*, 46(4):956–967.
- [Vels, 1990] Vels, A. (1990). Los mecanismos de defensa bajo el punto de vista psicoanalítico. *Agrupación de Grafoanalistas Consultivos de España. Bol*, 6:1–11.
- [Ventures, 1997] Ventures, D. (1997). Digevo ventures. <https://www.digevoventures.com/convocatorias/>. [Online; consultado el 20-Abril-2022].

- [Vivar, 2003] Vivar, D. M. (2003). *La mentira infantil: diagnóstico e intervención psicopedagógica*. Tesis doctoral, Universidad de Málaga.
- [Warmelink *et al.*, 2011] Warmelink, L., Vrij, A., Mann, S., Leal, S., Forrester, D., y Fisher, R. P. (2011). Thermal imaging as a lie detection tool at airports. *Law and human behavior*, 35(1):40–48.