ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA MECÁNICA

DESARROLLO DE UN ALGORITMO DE CLASIFICACIÓN AUTOMATICA DE GESTOS DE LA MANO

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGISTER EN MECÁTRONICA Y ROBÓTICA

ELVIS PATRICIO PAUCAR SOCASI

elvis.paucar@epn.edu.ec

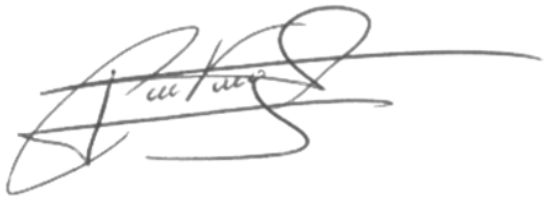
DIRECTOR: ING. WILLIAM RICARDO VENEGAS TORO, Ph.D

william.venegas@epn.edu.ec

Quito, marzo 2023

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Elvis Patricio Paucar Socasi, bajo mi supervisión.



Ing. William Ricardo Venegas Toro, Ph.D

DIRECTOR DE PROYECTO

DECLARACIÓN

Yo, Elvis Patricio Paucar Socasi , declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

Elvis Patricio Paucar Socasi

DEDICATORIA

A mi madre María Socasi, por su amor incondicional, su constante apoyo y por enseñarme a ser una persona fuerte, dedico con todo mi amor y gratitud. Gracias por ser mi luz en los momentos oscuros y por nunca dudar en mi capacidad. Tu sacrificio y dedicación son mi mayor motivación y me han llevado hasta aquí. Este trabajo es el resultado de mi esfuerzo y dedicación, pero también es un reflejo del amor y apoyo que he recibido de mi madre y toda mi familia. Espero que sea un paso mas hacia un futuro lleno de logros y satisfacciones.

AGRADECIMIENTOS

A Dios, por ser mi guía en todo momento y por darme la fuerza y sabiduría necesarias para seguir adelante, gracias por iluminar mi camino y por estar siempre presente en mi vida.

A todas las personas que han hecho posible la realización del presente trabajo. En primer lugar, quiero agradecer a mi tutor Ing. William Venegas, Ph.D por su apoyo, paciencia y guía en todo momento. A los profesores y personal que forman parte del programa de maestría de la Escuela Politécnica Nacional por compartir sus conocimientos y experiencias. A mi familia, en especial a mi madre, por su amor y apoyo incondicional durante todo el proceso.

Al proyecto de investigación PIS 20-04, como parte de una metodología para la evaluación de señales EMG.

CONTENIDO

RESUMEN

El objetivo del presente trabajo es desarrollar un algoritmo para la clasificación de gestos de la mano mediante el uso de señales mioeléctricas superficiales, el clasificador reconoce los gestos doble tap, fingers spread, fist, wave in, wave out y relax. Para lo cual se aplican técnicas de inteligencia artificial basadas en una red neuronal convolucional CNN. Para la realización del proyecto se utiliza la base de datos EMG- 120 de la Escuela Politécnica Nacional la cual contiene registros de señales mioeléctricas obtenidas mediante el dispositivo superficial myo armband ubicado en el antebrazo, de donde se generan imágenes de los gestos para poder almacenarlos es una estructura dataframe para luego preprocesar, entrenar el modelo de red neuronal usando la estructura ResNet como una red preentrenada realizando pruebas variando el número de capas de la red, se realizó la validación con el 20 % de la base de datos, el sistema se ha evaluado mediante métricas de desempeño de modelos de clasificación, obteniendo buenos resultados tanto con datos preprocesados como sin preprocesar. En la implementación del clasificador se hace uso de herramientas computacionales libres como el entorno de Google Colab, Pandas para la manipulación y análisis de datos, Fastai para el entrenamiento, validación del modelo.

PALABRAS CLAVE: Red Neuronal Convolucional, ResNet, Mioeléctrica, Google Colab.

ABSTRACT

The goal of the current work is to develop an algorithm for the classification of hand gestures through the use of superficial myoelectric signals, the classifier recognizes the gestures double tap, fingers spread, fist, wave in, wave out and relax. For which artificial intelligence techniques based on a CNN convolutional neural network are applied. To carry out the project, the EMG-120 database of the Escuela Politécnica Nacional is used, which contains records of myoelectric signals obtained through the superficial device myo armband located on the forearm, from which images of the gestures are generated to be able to store them. a dataframe structure to then preprocess, train the neural network model using the ResNet structure as a pretrained network, performing tests varying the number of layers of the network, validation was performed with 20 % of the database, the system was has been evaluated using classification model performance metrics, obtaining good results with both preprocessed and non-preprocessed data. In the implementation of the classifier, use is made of free computational tools such as the Google Colab environment, Pandas for data manipulation and analysis, Fastai for training, and model validation.

KEY WORDS: Convolutional Neural Network, ResNet, Myoelectric, Google Colab.

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

Se han propuesto varios modelos para clasificar los gestos de las manos utilizando señales mioeléctricas en diversas literaturas. Estos modelos tienen algunos problemas, como el uso de bases de datos privadas y recursos computacionales avanzados para entrenar las redes neuronales. Por ello, en este proyecto se propone **desarrollar un algoritmo de clasificación automática de gestos de la mano** utilizando señales mioeléctricas que son medidas en los músculos del antebrazo, y utilizando herramientas open source y técnicas de inteligencia artificial (Universidad de las Fuerzas Armadas ESPE, 2020).

Por ejemplo, este tipo de clasificador de gestos con las manos tiene múltiples aplicaciones en diversos campos técnicos y científicos, puede ser de gran ayuda en el avance de las prótesis biónicas de mano que son una opción muy deseada por las personas con alguna discapacidad, como una medida de sustitución al miembro faltante. Por otra parte, categorizar los gestos que la gente quiere hacer es problemático porque los patrones en las señales tienden a ser difíciles de discernir. Para ello, utilizamos técnicas estadísticas, algoritmos de programación avanzados o inteligencia artificial para clasificar gestos y reconocer patrones que describen el comportamiento de las señales. (Piña, 2017).

En la actualidad, la inteligencia artificial, en particular el aprendizaje profundo o Deep Learning, ha experimentado un avance significativo en la búsqueda de la inteligencia autónoma. Estas técnicas poseen la habilidad de aprender por sí mismas, utilizando información previa para generar decisiones y resultados precisos y consistentes de manera independiente (Muñoz, 2019).

Otro de los problemas en las diferentes investigaciones es el uso de software comercial, pues sus licencias tienen costos elevados que difícilmente pueden ser costeados por los presupuestos asignados a su investigación. Es por esto que se plantea en este trabajo hacer uso de las ventajas de Google Colab, es una herramienta ampliamente empleada en la comunidad de machine learning además, Google Colab te permite codificar y ejecutar código en Python en los servidores en la nube de Google, lo que te permite aprovechar la capacidad del hardware de Google, incluyendo las GPUs, sin importar la potencia de tu propio equipo. Lo único que se necesita es un navegador web (Colab, s.f.).

1.2 JUSTIFICACIÓN

En los últimos años, las señales mioeléctricas (EMG) se han utilizado ampliamente en el control de extremidades artificiales como prótesis, dispositivos médicos, interacción humano-computadora y otros campos. A medida que la inteligencia artificial y la tecnología robótica evolucionan, la intención de los movimientos de la mano humana se puede obtener mediante el uso de un algoritmo de inteligencia artificial para *analizar las señales EMG* recopiladas del brazo. La robótica y la inteligencia artificial se pueden aprovechar para ayudar mejorar a las personas discapacitadas a completar de forma independiente algunas interacciones básicas en su vida diaria. Las señales EMG, que no presentan estacionariedad, son una combinación de los potenciales de acción subcutáneos generados por la contracción muscular durante la actividad atlética (Asghari Oskoei & Hu, 2007). Además, es una de las principales señales físicas de un algoritmo inteligente para identificar la *clasificación de movimiento*. Este estudio tiene como objetivo utilizar la inteligencia artificial para *clasificar los gestos de la mano* a través de señales mioeléctricas, utilizando el entorno de Google Colab. Este entorno permite programar y ejecutar Python directamente en el navegador web, con la ventaja de tener acceso gratuito a GPUs (Colab, s.f.). También la de reducir costos por adquisición de licencias de software y buscar nuevas alternativas en el desarrollo y uso de estas tecnologías, aportando también al desarrollo de clasificación de gestos

para futuros prototipos de próstesis en el Ecuador, Colaboratory de Google (o simplemente Colab) es un servicio en línea basado en Jupyter Notebooks que tiene como objetivo difundir la educación y la investigación en el campo del aprendizaje automático. Ofrece un entorno de ejecución completamente configurado para el aprendizaje profundo, además de proporcionar acceso gratuito a una potente GPU (Carneiro y col., 2018).

La clasificación de señales EMG recopiladas de diferentes gestos es fundamental en las aplicaciones que utilizan estas señales como medio intermediario. Actualmente, la literatura sobre el reconocimiento de gestos o el control de extremidades artificiales mediante señales EMG se enfoca en extraer características tanto en el dominio del tiempo como en el de la frecuencia de dichas señales. El objetivo es distinguir las señales EMG mediante el reconocimiento de estas características de acuerdo con (Phinyomark y col., 2012).

Después de años de investigación, los expertos han propuesto algunas combinaciones de características que resultan efectivas tanto en el dominio del tiempo como en el de la frecuencia, y se han obtenido resultados prometedores utilizando los conjuntos de datos correspondientes. La elección de la extracción de características es particularmente importante porque los diferentes gestos se pueden distinguir mediante métodos tradicionales. Sin embargo, es difícil mejorar el rendimiento del reconocimiento de gestos basado en EMG por métodos tradicionales. Además, el proceso de diseño y selección de funciones puede ser complicado y las combinaciones de funciones son diversas, lo que lleva a un aumento de la carga de trabajo y resultados insatisfechos (Chen y col., 2020).

Para distinguir las señales EMG, se ha propuesto el uso de redes neuronales artificiales. Estos modelos pueden funcionar como codificadores automáticos para extraer características de forma automática, sin necesidad del proceso de extracción convencional. En los últimos años, el aprendizaje profundo (Deep Learning) ha tenido un gran éxito en el reconocimiento de imágenes (Chen y col., 2020).

En (Côté-Allard y col., 2017) se sugirió la idea de transformar la señal EMG en una

imagen, lo cual inspiró una nueva forma de abordar el análisis de dichas señales. Tratando las señales EMG originales como imágenes, se construyó un modelo de red neuronal para mejorar la precisión en la clasificación de estas señales.

Para los algoritmos de Deep Learning, la precisión de la prueba final se ve directamente afectada por el tamaño de datos de entrenamiento, pero no se puede esperar que un participante genere miles de ejemplos en un experimento durante el proceso de recopilación de datos. Sin embargo, es posible obtener una cantidad significativa de datos al combinar los registros de varios participantes y así formar una estructura de datos o Data Frame muy utilizada para el análisis de big data.

En términos generales, la clasificación de gestos de la mano a través de imágenes EMG instantáneas puede considerarse naturalmente como un problema de clasificación de imágenes que se puede abordar mediante el uso de inteligencia artificial y redes neuronales. Dado un conjunto de entrenamiento de imágenes capturadas y etiquetadas con gestos manuales realizados, se entrena un clasificador para predecir el gesto manual correspondiente a cada imagen EMG de entrada. Para abordar este problema de clasificación de imágenes EMG, se adoptó un enfoque de aprendizaje profundo.

Los modelos de clasificación de gestos de la mano tienen una amplia gama de aplicaciones en diversos campos científicos y tecnológicos, como la implementación de interfaces humano-máquina, el control de prótesis activas, la manipulación de imágenes durante procedimientos médicos, la generación de mapas de movimiento de la mano para interactuar con ambientes y objetos virtuales en aplicaciones de realidad aumentada o virtual, entre otros (Xu & Dai, 2017).

La investigación de la clasificación de gestos de la mano con IA es importante para el desarrollo de nuevas aplicaciones, y la inteligencia artificial es una fuente importante de innovación en este sentido. Es esencial fomentar prácticas académicas que contribuyan a la investigación y el desarrollo de tecnologías utilizando herramientas open source.

1.3 OBJETIVO GENERAL

Desarrollar un algoritmo de inteligencia artificial para clasificación automática de 5 gestos de la mano utilizando señales mioeléctricas mediante herramientas open source y técnicas de Deep Learning

1.4 OBJETIVOS ESPECÍFICOS

- Implementar en Google Colab el algoritmo de clasificación automática de 5 o más gestos de la mano con la utilización de señales mioeléctricas e inteligencia Artificial.

- Organizar una base de datos en un data frame de los gestos de la mano utilizando técnicas de Big Data.

- Entrenar una red neuronal artificial para la clasificación automática de los gestos de la mano utilizando técnicas de Deep Learning.

- Verificar la efectividad del algoritmo implementado utilizando métricas de Deep Learning.

1.5 MARCO TEÓRICO

1.5.1 Señales mioeléctricas

Las señales EMG son producidas por la actividad muscular del cuerpo humano durante la contracción o estiramiento muscular, y se adquieren mediante la técnica de electromiografía. En la Figura 1.1 se muestra un ejemplo de una señal EMG (Piña, 2017).

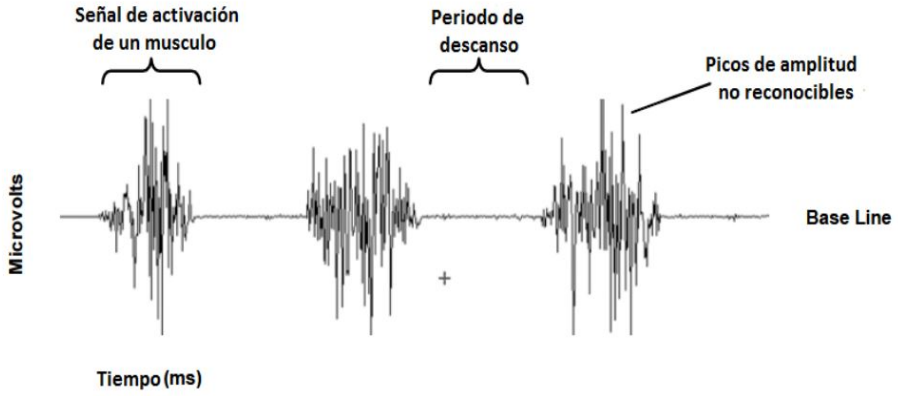


Figura 1.1: Señales EMG
Fuente: (Konrad, 2006)

La técnica de electromiografía se utiliza para medir las señales eléctricas generadas por las fibras musculares en respuesta a la activación del sistema nervioso. Las señales eléctricas registradas corresponden a los potenciales de campo generados por las corrientes subyacentes a los potenciales de acción de las fibras musculares, y se les conoce como electromiogramas (EMG). Un EMG típico obtenido durante una contracción muscular comprende los potenciales de campo sumados debido a la activación de muchas fibras musculares. El número de fibras musculares activadas durante una contracción muscular depende directamente de su intensidad y está relacionado con el número de neuronas motoras que se reclutan para la tarea. Se le llama unidad motora a una neurona motora y a las pocas cientos de fibras musculares que controla. El sistema nervioso regula la fuerza muscular ajustando la cantidad de actividad de las unidades motoras (Keenan & Enoka, 2012).

Las señales EMG son fáciles de registrar en la superficie de la piel que cubre los músculos y contienen información sobre sus movimientos particulares. Por lo tanto, se pueden analizar los patrones EMG para clasificar diversos movimientos debido a estas características (Hudgins y col., 1993).

Las señales mioeléctricas tienen un rango de amplitud en el orden de los mili voltios de alrededor de 10 mV pico a pico y comúnmente una frecuencia entre 6 Hz y 500 Hz, indicando una mayor frecuencia entre los rangos 20 Hz y 150 Hz

1.5.2 Aplicaciones señales mioeléctricas

Una señal EMG puede proporcionar información útil sobre el momento y la cantidad de activación muscular y, cuando se normaliza, una estimación de la fuerza ejercida por el músculo. Estas características son útiles para aplicaciones que van desde el control de dispositivos protésicos hasta la evaluación de estrategias utilizadas por el sistema nervioso para realizar diversas tareas.

1.5.2.1 Neurorehabilitación

La señal EMG se utiliza como señal de control para impulsar muchos tipos de dispositivos de neurorehabilitación, incluidos dispositivos protésicos, ortopédicos, asistidos por computadora, robóticos, exoesqueléticos, sillas de ruedas etc. El propósito de este enfoque es preservar, mejorar o reemplazar la función neuromotora. La información relacionada con el tiempo, la amplitud o el contenido de frecuencia de la señal EMG se puede utilizar como base para la señal de control. La señal derivada EMG se usa para controlar un motor mecánico para influir en la función neuromotora, o en combinación con estimulación eléctrica para influir directamente en la función muscular. Aunque las técnicas de un solo canal, como la amplitud de EMG y los tiempos de encendido/apagado, pueden tener éxito en sistemas de un solo músculo, el desarrollo de dispositivos realistas de neurorehabilitación exige el uso de enfoques de EMG multicanal con técnicas avanzadas de procesamiento de señales para proporcionar un control realista de los sistemas multidimensionales (Keenan & Enoka, 2012).

1.5.2.2 Biorretroalimentación

Hay una serie de herramientas disponibles para proporcionar información a las personas sobre la función fisiológica, incluidos los parámetros estimados a partir de la EMG de superficie. La aplicación de la EMG como herramienta de biorretroalimentación es común en los campos de la psicología clínica, la medicina

física y la rehabilitación, y el rendimiento deportivo. La razón típica de esta aplicación es aumentar la conciencia de la actividad muscular, a menudo con el objetivo de mejorar un desequilibrio debido a niveles inadecuados de activación muscular. Las fuentes de este desequilibrio en la actividad muscular pueden incluir debilidad, déficit postural, función refleja inadecuada, dolor crónico y estrés. El enfoque general en muchas aplicaciones de biorretroalimentación es identificar primero una cantidad inapropiada de actividad en un músculo y luego diseñar una intervención de entrenamiento que restaurará los niveles normales de activación muscular (Keenan & Enoka, 2012).

1.5.2.3 Ergonomía

La ergonomía se refiere a la mejora de las condiciones y requisitos laborales para adaptarse a las habilidades de los trabajadores en el lugar de trabajo. Una consideración importante en el diseño de entornos de trabajo es la carga física que las personas enfrentan a diario. Debido a la asociación entre la amplitud y la carga del EMG, aunque con algunas limitaciones, el EMG de superficie puede identificar y estimar las cargas experimentadas por partes específicas del sistema musculoesquelético. Esto es importante porque las cargas y fuerzas internas son difíciles de medir directamente (a diferencia de las fuerzas externas) y solo pueden estimarse mediante análisis biomecánicos. Por ejemplo, la EMG se usa habitualmente para estimar la cantidad de compromiso muscular en diferentes tareas y para evaluar la posible etiología de los problemas en el lugar de trabajo, incluidos los trastornos musculoesqueléticos, el estrés y el aumento de la tensión muscular, el dolor y la fatiga muscular (Keenan & Enoka, 2012).

1.5.2.4 EMG clínico

Las señales EMG intramusculares y de superficie se utilizan como herramientas de diagnóstico para los trastornos neuromusculares. Los electrodos de aguja y alambre fino, por ejemplo, se pueden insertar en el músculo que se está investigando

para cuantificar diferentes parámetros de las señales EMG. Debido a la mayor selectividad de este tipo de registro EMG, se pueden inferir detalles sobre las propiedades de las fibras musculares y las unidades motoras. Los parámetros más utilizados en la EMG de diagnóstico, además de las estimaciones de la velocidad de conducción nerviosa, comprenden la magnitud y la duración del potencial de acción generado por la unidad motora, los giros, los cruces por cero y el número de fases. Cada uno de estos parámetros puede proporcionar información útil al médico experto en el diagnóstico de miopatías y neuropatías. Además de estos enfoques invasivos, la EMG de superficie se utiliza habitualmente para la evaluación de trastornos del movimiento, incluidos temblores, mioclonías, distonía, discinesia y anomalías de la marcha (Keenan & Enoka, 2012).

1.5.3 Myo Armband

Este dispositivo está diseñado para detectar gestos musculares y se coloca en el antebrazo. Cuenta con ocho sensores EMG, un giroscopio de tres ejes (roll, pitch, yaw), un acelerómetro y un magnetómetro, lo que le permite detectar el movimiento en cualquier dirección. El procesador incorporado en el dispositivo interpreta la actividad muscular y la lectura de los movimientos. Además, cuenta con una conexión inalámbrica para transmitir los datos procesados del EMG a computadoras y dispositivos móviles. Estos datos se pueden utilizar para estudiar las diferentes actividades musculares y su activación en las extremidades superiores.

El brazalete tiene la capacidad de registrar señales electromiográficas a una frecuencia de 200Hz y codifica cada medición en 8 bits. Es importante destacar que cuenta con un sistema de amplificación y filtrado de señal integrado en su estructura para poder capturar estas señales con precisión (Flores, 2021).

Figura 1.2: Myo Armband
Fuente: (Castello, 2017)

1.5.4 Clasificador

Un clasificador es un algoritmo de aprendizaje automático que se encarga de etiquetar o clasificar una nueva muestra de datos. Este tipo de sistema aprende a reconocer patrones en un conjunto de datos etiquetados y posteriormente aplica lo aprendido para predecir la etiqueta de nuevos datos. Un ejemplo común de clasificación es la detección de correo no deseado, donde el clasificador aprende a identificar los mensajes no deseados mediante el análisis de patrones en los datos previamente etiquetados. Además, los clasificadores se utilizan en diferentes aplicaciones, tales como la detección de objetos en imágenes, la clasificación de texto y el análisis de sentimientos.

1.5.5 Tipos de clasificadores

Hay diversos tipos de clasificadores, los cuales se detallan a continuación:.

1.5.5.1 Clasificadores paramétricos

Son aquellos que asumen una distribución específica para los datos, como una distribución normal, funciones de densidad de probabilidad. Estos clasificadores tienen un número finito de parámetros que se pueden ajustar mediante el entrenamiento. Ejemplos incluyen el clasificador bayesiano, máxima probabilidad,

clasificador basado en reglas.

1.5.5.2 Clasificadores no paramétricos

Evitan el uso de medidas estadísticas para llevar a cabo la clasificación, no asumen ninguna distribución específica para los datos y su número de parámetros tienden a ser infinitos. Estos clasificadores son más flexibles y pueden adaptarse mejor a datos complejos o no estructurados. Ejemplos incluyen el clasificador basado en vecinos más cercanos (k-NN), red neuronal artificial, clasificador basado en árboles de decisión.

En general, los clasificadores paramétricos son más rápidos y fáciles de entrenar, pero pueden ser menos precisos en casos de datos complejos. Los clasificadores no paramétricos son más precisos pero pueden ser más lentos y difíciles de entrenar.

1.5.6 Inteligencia Artificial

La inteligencia artificial (IA) es un campo de la informática que se enfoca en desarrollar sistemas informáticos capaces de realizar tareas que requieren habilidades y conocimientos asociados con la inteligencia humana, como la detección de patrones, el aprendizaje y la toma de decisiones. La IA se puede clasificar en diferentes categorías, entre las cuales se encuentran la IA débil, que se concentra en una tarea específica, y la IA fuerte, que busca emular la inteligencia humana en general.

Existen diversas aplicaciones de la IA, tales como el aprendizaje automático, la robótica ,el procesamiento del lenguaje natural. Los progresos en la IA han dado lugar a la elaboración de sistemas que pueden llevar a cabo labores complicadas, como el reconocimiento de imágenes y voz, lo que ha tenido una repercusión relevante en diversos ámbitos, tales como la medicina, la fabricación y los servicios. No obstante, la IA también ha generado preocupaciones éticas y de seguridad, y es necesario implementar una regulación adecuada para abordar estos problemas.

En definitiva, el desarrollo de la inteligencia artificial es un proceso histórico que ha evolucionado con el tiempo como se puede apreciar en la Figura 1.3.

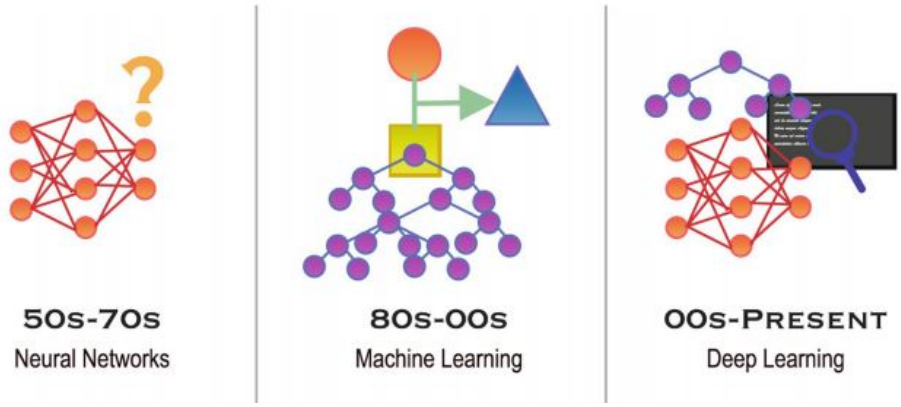


Figura 1.3: Linea de tiempo de la IA
Fuente: (Yalçn, 2021)

1.5.6.1 Machine Learning

El Aprendizaje Automático o Machine Learning es una rama de la Inteligencia Artificial que se enfoca en estadísticas, matemáticas y técnicas lógicas para extraer patrones (es decir, información) a partir de un conjunto de datos utilizado para el entrenamiento y aplicar las inferencias a datos no vistos. Nuevamente, estos avances recientes en ML fueron posibles gracias a la nueva era de big data y el gran avance en la capacidad computacional. Es importante destacar que ML se diferencia de otras formas de IA en que no requiere una programación extensa y complicada, sino que tiene la capacidad de aprender patrones y luego aplicarlos. Por lo tanto, ML no necesita considerar todas las soluciones posibles (es decir, ser determinista) y puede gestionar el ruido y la incertidumbre (Iman y col., 2020).

1.5.6.2 Deep Learning

El aprendizaje profundo, también conocido como deep learning (DL), es una rama del machine learning que se enfoca en el uso de múltiples capas de neuronas para identificar patrones y características de datos sin procesar. Estas capas interconectadas de neuronas forman redes neuronales artificiales (ANN) como se

ilustra en la Figura 1.4, que es un algoritmo diseñado para simular el funcionamiento del cerebro humano. Existen diferentes tipos de redes neuronales artificiales para diferentes propósitos. En resumen, los algoritmos de aprendizaje profundo son una subcategoría de los algoritmos de machine learning (Yalçn, 2021).

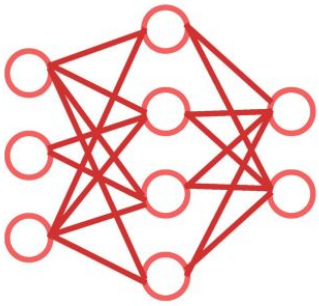


Figura 1.4: Red Neuronal Artificial
Fuente: (Yalçn, 2021)

Los cuatro enfoques del aprendizaje automático, que son el supervisado, semisupervisado, no supervisado y reforzado, también pueden aplicarse al aprendizaje profundo. En casos en los que hay grandes cantidades de datos y suficiente capacidad de procesamiento, el aprendizaje profundo suele ser superior a otros algoritmos de aprendizaje automático. En particular, los algoritmos de aprendizaje profundo son altamente efectivos en el procesamiento de imágenes, el reconocimiento de voz y la traducción automática.

1.5.6.3 Data Science

La ciencia de datos se compone de tres áreas principales: la recopilación, el almacenamiento y el análisis de datos, ya sean estructurados o no estructurados. La mejora de los métodos de recopilación de datos se ha visto impulsada por la rápida expansión de Internet de alta velocidad en todo el mundo, que ha dado lugar a conexiones inalámbricas más económicas y una amplia variedad de sensores electrónicos más asequibles, como relojes inteligentes, rastreadores de ejercicio y cámaras. El almacenamiento de datos ha mejorado gracias a la nube, lo que ha permitido una mayor recopilación de datos a bajo costo para una proporción cada vez mayor de la población.

El análisis de datos consta de dos componentes principales: preprocesamiento y procesamiento. El preprocesamiento se refiere a varios aspectos de la gestión de datos sin procesar, incluidos datos desequilibrados, técnicas de imputación de datos faltantes, detección y tratamiento de valores atípicos y procedimientos de etiquetado de datos. El procesamiento se refiere a la extracción de información y conocimiento de datos preprocesados para identificar patrones, hacer predicciones y/o clasificar datos (Iman y col., 2020).

1.5.6.4 Big Data

Big data es el término que se utiliza para describir la captura, almacenamiento, análisis, búsqueda, intercambio, visualización y actualización de grandes cantidades de datos de manera eficiente (Yalçn, 2021). Estos datos se caracterizan principalmente de las siguientes tres maneras (Silaparasetty, 2020b):

- Alto volumen: Esto hace referencia a la magnitud de la cantidad de datos que son producidos y guardados. La cantidad de estos datos es inmensa, ya que encuentra sus fuentes en imágenes, videos, audios y texto.
- Alta velocidad: Hace referencia a la rapidez con la que los datos son generados y procesados. Por lo general, estos datos están disponibles en tiempo real, lo que significa que se producen y manejan continuamente.
- Alta veracidad: Hace referencia a la fiabilidad y precisión de los datos, ya que la variabilidad en la calidad de los datos puede tener un impacto significativo en los resultados del análisis.

Algunos de los desafíos que enfrenta Big Data incluyen los siguientes (Silaparasetty, 2020b):

- Recopilación de datos: Recopilar los datos no es tarea fácil debido a la gran cantidad de información disponible
- Almacenamiento de datos: Se requieren unidades de almacenamiento muy potentes para almacenar cantidades tan grandes de datos.

- Transferir y compartir datos: Para transferir y compartir con éxito grandes cantidades de datos, se requieren técnicas y herramientas avanzadas.

1.5.6.5 DataFrame

Un DataFrame es una estructura de datos tabulares muy similar a una hoja de cálculo. Esta estructura de datos está diseñada almacenar y manipular matrices 2-D. De hecho, Un dataframe se compone de una serie de columnas organizadas, en las que cada una puede contener un valor de tipo distinto (como numérico, de texto, booleano, entre otros). A diferencia de las series, que tienen una matriz de índice que contiene etiquetas asociadas con cada elemento, el dataframe tiene dos matrices de índice. La primera matriz de índice, asociada a las líneas, tiene funciones muy similares a la matriz de índice en serie. De hecho, cada etiqueta está asociada con todos los valores de la fila. La segunda matriz contiene una serie de etiquetas, cada una asociada con una columna en particular. Otra manera de explicar un dataframe es que se puede ver como un diccionario de series, en el cual las claves representan los nombres de las columnas y los valores son las series que forman cada columna del dataframe. Además, todos los elementos de cada serie se asignan de acuerdo con una matriz de etiquetas, denominada índice (Nelli, 2018).

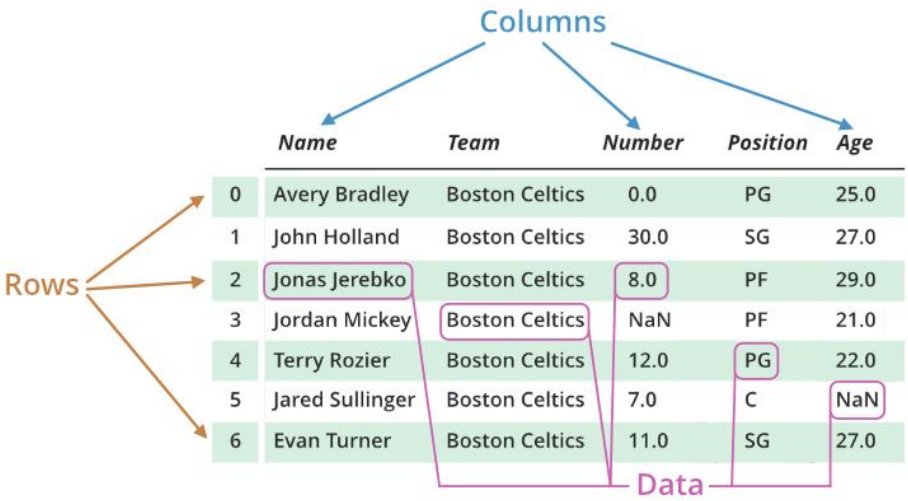


Figura 1.5: Estructura de un DataFrame.
Fuente: (Nelli, 2018)

Esta estructura de datos es la mas utilizada para realizar análisis de big data, para

la creación del dataframe se utiliza la librería es una herramienta de código abierto para el análisis de datos que ofrece una estructura de datos sencilla y diseñada específicamente para facilitar el manejo de datos en Python. Pandas lee datos de varios formatos de archivo (por ejemplo, CSV, bases de datos SQL y Parquet) y crea un objeto de Python, DataFrame, con filas y columnas similares a una tabla en Excel. Pandas se puede integrar con herramientas de visualización científica como los cuadernos Jupyter; Los cuadernos de Jupyter proporcionan una interfaz unificada para organizar, ejecutar código y visualizar resultados sin consultar los detalles de los sistemas de bajo nivel. El amplio conjunto de funciones que están disponibles en Pandas lo convierte en una herramienta de análisis de datos muy popular de la actualidad. Sin embargo, su limitación radica en la escalabilidad. Pandas no proporciona almacenamiento de datos ni soporte para interactuar con datos distribuidos, ya que se ha centrado en el cálculo en memoria en un solo nodo. Otra limitación conocida de Pandas es su consumo de memoria. Esto se debe a los requisitos de memoria interna subyacentes sobre los cuales el creador de Pandas, McKinney, aconsejó: "debe tener de 5 a 10 veces más RAM que el tamaño de su conjunto de datos"(Sinthong & Carey, 2019).

1.5.6.6 Diagrama de Taxonomía

El diagrama de taxonomía, presentado en la Figura 1.6, permite visualizar la relación entre los términos adyacentes.

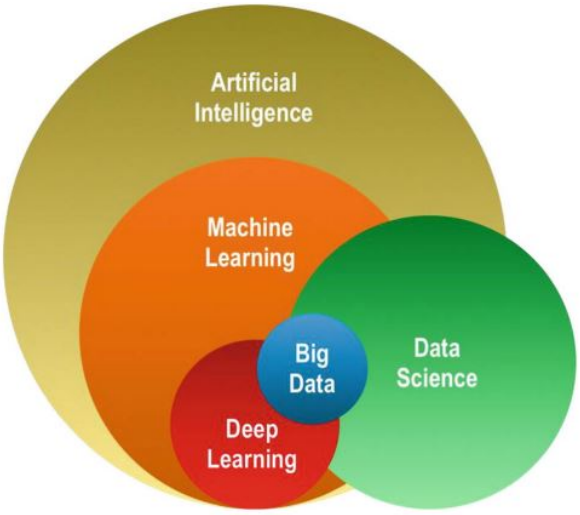


Figura 1.6: La taxonomía de la inteligencia artificial y la ciencia de datos
Fuente: (Yalçn, 2021)

La taxonomía presentada es una prueba clara de por qué existe la ambigüedad. El aprendizaje profundo, el aprendizaje automático y la inteligencia artificial están estrechamente relacionados y a menudo se mencionan juntos. Además, un proyecto de aprendizaje profundo puede denominarse como un proyecto de ciencia de datos o big data, lo que puede generar confusión. Aunque estas prácticas de denominación no son incorrectas, es importante comprender las áreas en las que se superponen y las que no para evitar confusiones (Yalçn, 2021).

1.5.6.7 Redes Neuronales

La idea detrás de la red neuronal o red neuronal artificial se basa en la estructura y funcionamiento de las redes neuronales biológicas. Al igual que el cerebro humano, estas redes son capaces de aprender y realizar tareas específicas sin necesidad de una programación explícita. Una red neuronal está compuesta por múltiples neuronas interconectadas, que forman una red, de ahí su nombre. La neurona artificial es la unidad básica de una red neuronal, y es una función matemática que imita el funcionamiento de las neuronas en el cerebro humano. La Figura 1.7 ilustra este concepto (Silaparasetty, 2020a).

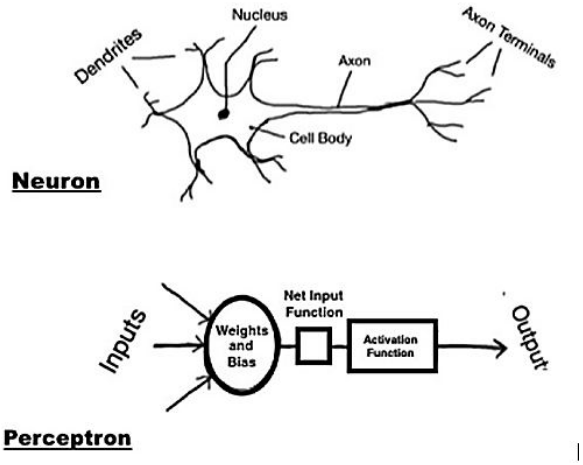


Figura 1.7: Una neurona biológica y una neurona artificial
Fuente: (Silaparasetty, 2020a)

1.5.6.8 Red Neuronal Artificial

Basándose en la inspiración de la neurona biológica, la red neuronal artificial (ANN) es una sociedad de agentes conexionistas que aprenden y transfieren información de una neurona artificial a otra. A medida que se transfieren datos entre neuronas, se aprende una jerarquía de representaciones o una jerarquía de características, de ahí el nombre de aprendizaje profundo de representaciones o aprendizaje profundo (Bisong, 2019b).

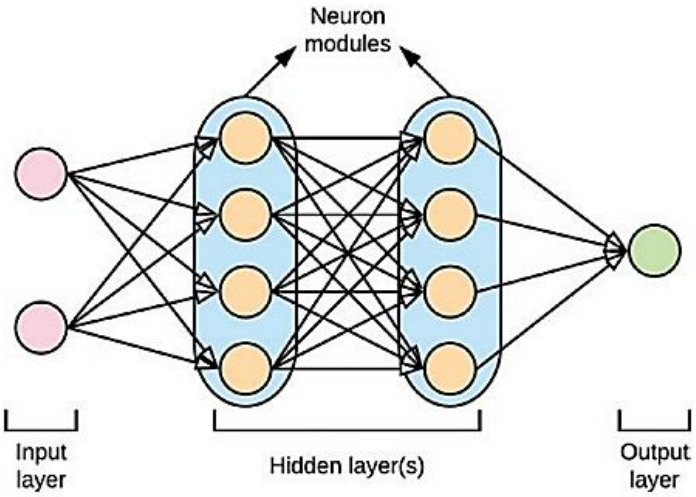


Figura 1.8: Arquitectura ANN
Fuente: (Bisong, 2019b)

En términos generales, se puede dividir una red neuronal artificial en tres partes llamadas capas, que son conocidas como:

- Capa de entrada: La función de la capa de entrada es tomar información, señales, características o mediciones del entorno externo y procesarlas como entradas en la red neuronal artificial. Los valores de entrada se normalizan dentro de los límites establecidos por las funciones de activación, lo que mejora la precisión numérica para las operaciones matemáticas posteriores en la red.

- Capas ocultas, intermedias o invisibles: Las capas mencionadas están formadas por un conjunto de neuronas que se dedican a detectar patrones relacionados con el proceso o sistema en cuestión. Es en estas capas donde se lleva a cabo la mayor parte del procesamiento interno de la red.

- Capa de salida: Las neuronas en esta capa final también están interconectadas y su función es generar y entregar las salidas finales de la red neuronal, las cuales se derivan del procesamiento de información realizado por las capas anteriores.

Podemos clasificar las principales arquitecturas de las redes neuronales artificiales según la disposición de las neuronas, su interconexión y la composición de sus capas. En este sentido, se pueden identificar cuatro tipos de arquitecturas principales: (i) redes feedforward de una sola capa, (ii) redes feedforward multicapa, (iii) redes recurrentes y (iv) redes en malla. (da Silva y col., 2017).

1.5.6.9 Redes Neuronales Convolucionales

Las CNN se diferencian de las ANN convencionales principalmente en su aplicación en el reconocimiento de patrones dentro de imágenes. Las características específicas de la imagen se pueden codificar en la arquitectura, lo que hace que la red sea más adecuada para tareas de imagen y reduce los parámetros necesarios para configurar el modelo. La complejidad computacional requerida para calcular

datos de imagen es una limitación de las ANN tradicionales (O'Shea & Nash, 2015). Las capas principales de las redes neuronales convolucionales se dividen en tres tipos distintos:

- Capa convolucional
- Capa de pooling
- Capa full connected o totalmente conectada

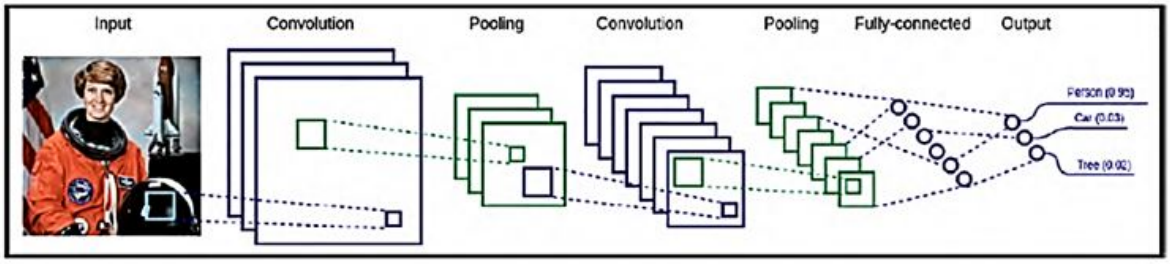


Figura 1.9: Configuración estándar de una CNN
Fuente: (Pajankar & Joshi, 2022)

Capa de convolución

La capa de convolución contiene filtros y mapas de características que se utilizan para aplicar la convolución sobre los píxeles de la imagen de entrada y detectar características específicas.

La convolución es el proceso mediante el cual se aplica una función o producto escalar a una matriz para extraer información específica de la matriz. La función se implementa como una ventana deslizante a través de la matriz, y es más conocida como filtro convolucional o kernel (Bisong, 2019a).

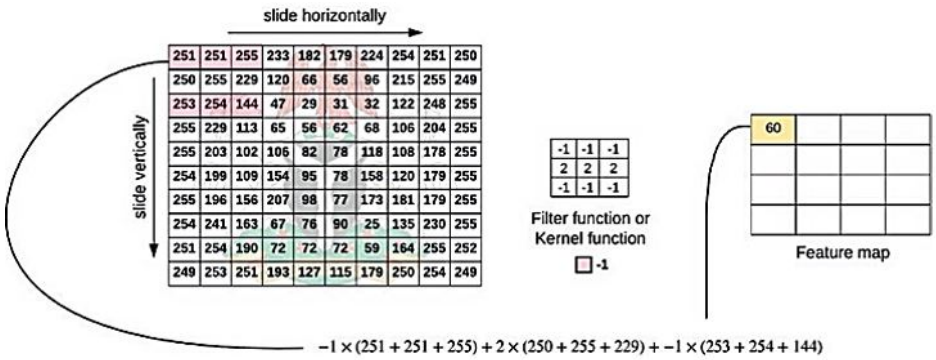


Figura 1.10: Operación de convolución
Fuente: (Bisong, 2019a)

Capa de pooling

La función de la capa de pooling es disminuir la muestra del mapa de características generado por la capa convolucional para lograr una representación más compacta. En otras palabras, la capa de pooling resume o sintetiza las características relevantes de la imagen aprendidas en las capas anteriores de la red. Al hacerlo, también ayuda a evitar que la red se sobreajuste. Además, la reducción en el tamaño de entrada también es un buen augurio para los costos de procesamiento y memoria al entrenar la red. Las funciones de agregación realizadas por la capa de agrupación incluyen máximo, suma y promedio. La función de agregación utilizada con más frecuencia en la práctica es max o maxpooling. Las funciones de agregación de la capa de pooling funcionan como filtros para las capas de la red, tal como se ilustra en la Figura 1.11 (Bisong, 2019a).

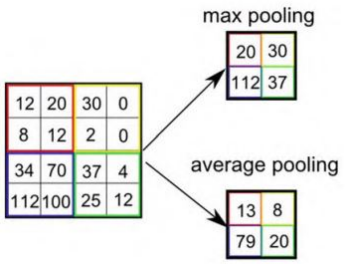


Figura 1.11: Operación de pooling
Fuente: (Teoh & Rong, 2022)

Capa full connected

La capa de red totalmente conectada (FCN) es nuestra red neuronal feedforward habitual o perceptrón multicapa. Estas capas suelen tener una función de activación no lineal. En todo caso, la FCN es la última capa de la red neuronal convolucional y utiliza una activación softmax para producir las probabilidades de que una entrada pertenezca a una clase específica.

Antes de pasar una entrada al FCN, la matriz de la imagen deberá aplanarse. Por ejemplo, una matriz de imagen de 28 x 28 x 3 se convertirá en 2352 pesos de entrada más un sesgo de 1 en la red totalmente conectada. En las redes convolucionales, antes de calcular las probabilidades finales mediante la función softmax en la capa FCN, los mapas de características de la capa convolucional o de pooling se convierten en una matriz unidimensional. Este proceso se conoce como aplanamiento (Bisong, 2019a).

1.5.7 Cloud Computing

Otra forma de referirse a la çomputación en la nube.[es] mediante el término çloud computing.[es] la práctica en la que los servicios informáticos, como las opciones de almacenamiento, las unidades de procesamiento y las capacidades de red, están expuestos para el consumo de los usuarios a través de Internet (la nube). Estos servicios van desde la facturación gratuita hasta el pago por uso.

La idea central detrás de la computación en la nube es hacer que la potencia computacional agregada esté disponible para el consumo a gran escala. Al hacerlo, el principio microeconómico de las economías de escala entra en vigencia donde a medida que la escala de las operaciones aumenta, se reduce el costo por unidad de producción. En un contexto de cloud computing, las empresas o los individuos pueden aprovechar la misma velocidad y potencia de los servicios de computación agregados de alto rendimiento y pagar solo por lo que usan y renunciar a estos recursos informáticos cuando ya no los necesiten.

El concepto de computación en la nube había existido como sistemas de tiempo compartido desde los primeros años de la computadora moderna donde los trabajos enviados por diferentes usuarios estaban programados para ejecutarse en un mainframe. La idea de las máquinas de tiempo compartido se desvaneció con la llegada de la PC. Ahora, con el auge de los centros de datos empresariales administrados por grandes empresas de TI como Google, Microsoft, Amazon, IBM y Oracle, la noción de computación en la nube ha resurgido con el giro adicional de la tenencia múltiple en lugar del tiempo compartido. Este modelo informático está configurado para alterar la forma en que trabajamos y utilizamos los sistemas y servicios de software (Bisong, 2019c).

1.5.7.1 Google Colaboratory

Una creación de Google Research es Google Colaboratory o Google Colab que permite programar en lenguaje Python y ejecutarlo desde un navegador web, dando acceso gratuito a recursos computacionales adecuados para el trabajo con aprendizaje automático, ciencia de datos, educación entre otros (Colab, s.f.).

Colab provee de recursos computacionales que pueden incluir una GPU o TPU, pero su uso es limitado por el tiempo de acceso debido a sus políticas de uso con el objetivo de ofrecer un entorno interactivo, mientras se haga uso de más recursos menor tiempo de acceso gratuito se tendrá. Existe la opción de suscribirse a Cobab Pro o Colab Pro+ lo cual permite usar mejores recursos y por algo más de tiempo, sin embargo, no se garantiza un tiempo de acceso en específico. Y una tercera opción de pago que permite acceder a máquinas virtuales de Google Colab, gestionar los recursos de hardware requeridos y utilizar el entorno de forma persistente. A continuación, en la Tabla 1.1 se indica los recursos computacionales que se pueden encontrar en el entorno de Colab (Tapia, 2022).

GPU	GPU RAM	CPUs	RAM
K80	12 GB	2 CPU	13 GB
T4	16 GB	2 CPU	13 hasta 25 GB
P100	16 GB	2 CPU	13 hasta 25 GB
V100	16	16 GB	hasta 52 GB

Tabla 1.1: Recursos de hardware disponibles en Google Colab
Fuente: (Tapia, 2022)

1.5.8 Métricas de rendimiento

Las medidas de rendimiento son esenciales en la clasificación para comparar algoritmos de Machine Learning y Deep Learning y determinar el mejor en función del objetivo de investigación (Borja-Robalino y col., 2020).

1.5.8.1 Matriz de confusión

Se utiliza una matriz de confusión para resumir el desempeño de un clasificador en relación a los datos de prueba. Esta matriz es bidimensional y se indexa por la verdadera clase de un objeto y la clase asignada por el clasificador. Es una herramienta importante para evaluar el rendimiento de un clasificador en problemas de clasificación (Ting, 2010).

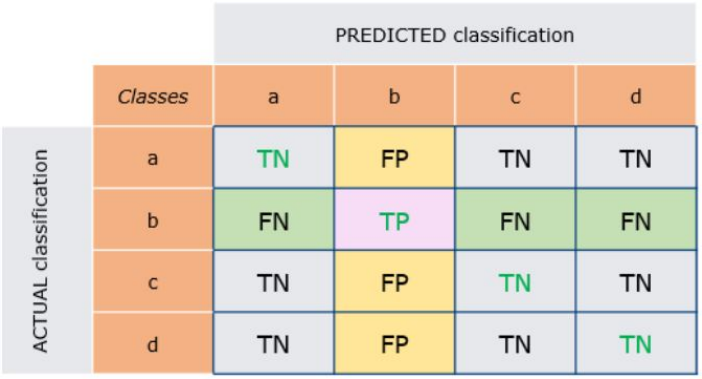


Figura 1.12: Matriz de confusión
Fuente: (Grandini y col., 2020)

La matriz tiene celdas designadas como verdaderos positivos (TP), verdaderos

negativos (TN), falsos positivos (FP) y falsos negativos (FN), como se ilustra en la Figura 1.12.

TP: Número de objetos positivos que fueron clasificados correctamente.

TN: Número de objetos negativos que fueron clasificados correctamente.

FN: Número de objetos positivos que fueron clasificados incorrectamente como negativos.

FP: Número de objetos negativos que fueron clasificados incorrectamente como positivos.

Con estos valores se puede determinar parámetros que permiten evaluar el desempeño del modelo.

1.5.8.2 Exactitud (Accuracy)

La exactitud o accuracy es una de las métricas más populares en la clasificación de clases múltiples es la probabilidad de que la predicción del modelo sea correcta, se calcula con la Ecuación 1.1.

$$Accuracy = \frac{TN + TP}{TP + TN + FP + FN} \tag{1.1}$$

La exactitud devuelve una medida general de cuánto predice correctamente el modelo en todo el conjunto de datos.

1.5.8.3 Precisión

La precisión expresa la proporción de unidades que nuestro modelo dice que son positivas y que en realidad son positivas. En otras palabras, la precisión nos dice cuánto podemos confiar en el modelo cuando predice que un individuo es positivo , se calcula con la Ecuación 1.2.

$$Precision = \frac{TP}{TP + FP} \tag{1.2}$$

1.5.8.4 Recall o sensibilidad

El recall es una métrica que indica la proporción de casos positivos que fueron identificados correctamente en un conjunto de datos. Se obtiene utilizando la ecuación 1.3.

$$Recall = \frac{TP}{TP + FN} \tag{1.3}$$

1.5.8.5 F1 Score

Esta medida determina la eficacia global de la precisión y la sensibilidad, mediante la media armónica que es útil para encontrar la mejor compensación entre las dos cantidades, se calcula con la Ecuación 1.4.

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{1.4}$$

2 METODOLOGÍA

En la sección siguiente se explica la metodología empleada en la elaboración del proyecto:

- Recopilación de datos:La fase inicial consiste en obtener los datos electromiográficos (EMG) de los movimientos realizados por la mano.
- Limpieza y preprocesamiento de datos: Después de la recolección de los datos, es fundamental realizar una fase de limpieza y preprocesamiento para eliminar cualquier ruido o distorsión y garantizar que estén en un formato apropiado para su posterior análisis.
- División de datos: Después de haber preprocesado los datos, se procede a realizar la división de los mismos en dos grupos o conjuntos distintos: un conjunto de entrenamiento y otro de prueba.
- Selección de la arquitectura de red neuronal: Se utiliza una arquitectura de Deep Learning, para crear la red neuronal que será utilizada para la clasificación.
- Entrenamiento de la red neuronal: Durante el entrenamiento de la red neuronal, se utilizan los datos del conjunto de entrenamiento para ajustar los pesos y los sesgos con el fin de mejorar el rendimiento en la tarea de clasificación.
- Evaluación de la red neuronal: Se evalúa el desempeño de la red neuronal en la tarea de clasificación utilizando el conjunto de prueba.
- Uso de la red neuronal para clasificación automática de gestos de la mano:

Una vez entrenada y evaluada la red neuronal, se puede utilizar para clasificar automáticamente nuevos gestos de la mano.

La Figura 2.1 ilustra la metodología empleada en este estudio.

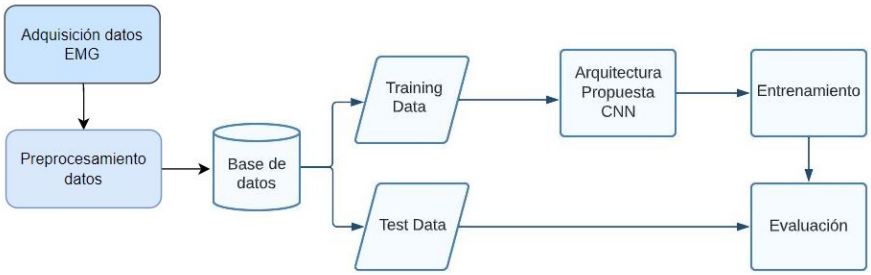


Figura 2.1: Metodología de clasificación de gestos.
Fuente: autor

En cuanto a la programación se la realizará en el entorno de Google Colaboratory que permite manejar y ejecutar código en lenguaje Python y puede realizar el procesamiento a través de una GPU.

2.1 ENTORNO DE GOOGLE COLAB

En la Figura 2.2 se presenta una secuencia que muestra cómo iniciar un entorno en Colab. Se puede encontrar información adicional en el Anexo 6.1.

Figura 2.2: Diagrama inicio Gooble Colab.
Fuente: autor

2.2 BASE DE DATOS O DATASET

La base de datos es un componente crucial para el éxito del proyecto, ya que se trabaja con un sistema inteligente cuyo objetivo es clasificar los gestos a partir de señales mioeléctricas. Los datos de las señales EMG presentan patrones difíciles de identificar y tienden a variar aleatoriamente entre personas. Por ejemplo, una señal adquirida de una persona en reposo es diferente a una señal adquirida de la misma persona realizando alguna actividad física, como caminar o correr, debido a los cambios biológicos asociados con esas actividades. El proceso utilizado para crear la base de datos se muestra en la Figura 2.3.

Figura 2.3: Proceso de generación de una estructura de datos tipo DataFrame.
Fuente: autor

2.2.1 Adquisición de datos

Para el presente trabajo se utilizó el dataset EMG-EPN-120 creado por el Laboratorio de Investigación en Inteligencia y Visión Artificial, 2020 de la Escuela Politécnica Nacional, este dataset o conjunto de datos contiene mediciones de las señales mioeléctricas (EMG) de 60 usuarios para training y 60 para testing . Las mediciones se realizaron utilizando el sensor de pulsera comercial Myo Armband. Los datos se subdividen en dos conjuntos: uno para entrenamiento y otro para pruebas. Los datos del usuario contienen información sobre la edad, el sexo, etc. Además, se incluyen señales de EMG con 260 repeticiones de cada usuario (10 repeticiones para el gesto de relajación y 50 para los otros 5 gestos). La Figura 2.4 muestra los gestos incluidos en el conjunto de datos: doble tap (pellizcar), wave out (hacia afuera), wave in (hacia adentro), fingers Spread (abrir), fist (puño) y el gesto de relajación.

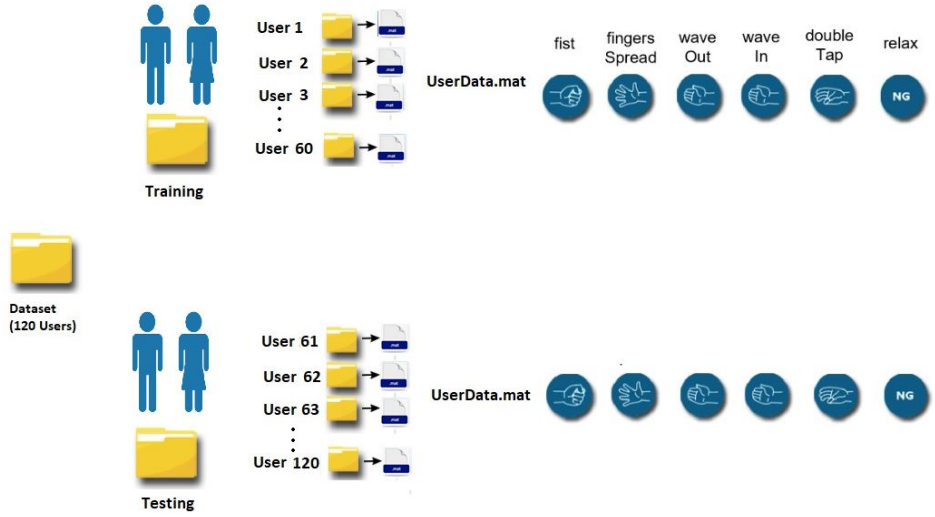


Figura 2.4: Dataset EMG EPN 120.
Fuente: autor

Los datos EMG se adquirieron mediante 8 sensores superficiales colocados en el antebrazo de los participantes y se registraron a una frecuencia de muestreo de 200 Hz con 8 bits de resolución por sensor. Cada muestra de datos tiene una duración de 5 segundos. Una ilustración del Myo sensor de brazalete y los cinco gestos con las manos se presentan en la Figura 2.5.

(a) Ubicación del sensor

(b) Gestos que reconoce

Figura 2.5: Brazalete MyoArmband

Para comenzar el desarrollo del sistema de clasificación en este proyecto inicialmente el dataset debe estar importado en google drive, al tener una extensión de archivo .mat se deben utilizar librerías como son scipy.io para acceder a su información desde google colab, para luego proceder a tomar los datos de interés para nuestro trabajo los cuales son los datos de lecturas de las señales mioeléctrica, posteriormente, realizar un preprocesamiento y generar graficas de la señal y guardarlos en las respectivas carpetas, ademas de la etiqueta del gesto a clasificar.

En la Figura 2.6 se importa las librerías necesarias junto con las direcciones para poder cargar los datos del dataset a google colab.

```
[ ]
    import os # damos acceso a python a nuestro almacenamiento interno
    import matplotlib.pyplot as plt #graficar
    import scipy.io #conectar con matlab

[ ] direccion_proyecto="/content/drive/MyDrive/Colab Notebooks/2020_EMG_EPN_120/user1/"     # direccion en drive del dataset
    print(direccion_proyecto)

[ ] direccion_mat=os.path.join(direccion_proyecto,"userData.mat")    # direccion del archivo .mat de cada usuario
    print(direccion_mat)

[ ] datos=scipy.io.loadmat(direccion_mat)     # carga del archivo .mat al entorno Colab
```

Figura 2.6: Código para cargar datos a google colab.
Fuente: autor

En la Figura 2.7 muestra el código utilizado para acceder a los datos de señales mioeléctricas del gesto con su respectiva etiqueta, también se puede observar que se tiene una matriz donde se guardan las lecturas de los ocho sensores que tiene el brazalete myoarmband, se debe tener en cuenta que para cada señal de gesto y etiqueta se debe acceder con una diferente ruta o dirección.

```
[ ] datos['userData'][0][0][2][0][0][0][0][0][0]

    array([[ 0.0078125, -0.0234375, -0.0390625, ..., -0.109375 , -0.0234375,
            -0.0078125],
           [ 0.       , -0.0390625,  0.0390625, ...,  0.09375  ,  0.03125  ,
            -0.0078125],
           [-0.0234375, -0.0390625, -0.0390625, ..., -0.015625 ,  0.0078125,
            -0.015625 ],
           ...,
           [ 0.       ,  0.       ,  0.       , ..., -0.046875 ,  0.0546875,
            -0.0078125],
           [-0.0234375, -0.0078125, -0.0078125, ...,  0.2734375,  0.0625   ,
            -0.0078125],
           [ 0.       , -0.03125  , -0.0078125, ..., -0.015625 ,  0.0078125,
            -0.0234375]])

[ ] datos['userData'][0][0][2][0] [0][0][0][3][0]

    'fist'
```

Figura 2.7: Código para acceder a datos de usuario.
Fuente: autor

Posteriormente como se indica en la Figura 2.8 se puede representar gráficamente la señal mioeléctrica del gesto adquirido del dataset.

```
[ ]  plt.plot(datos['userData'][0][0][2][0][0][0][0][0])
     plt.ylim(-1.5, 1.5)
```

```
(-1.5, 1.5)
```

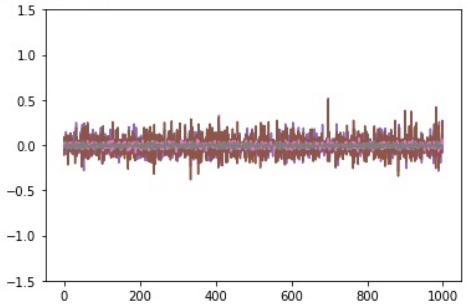
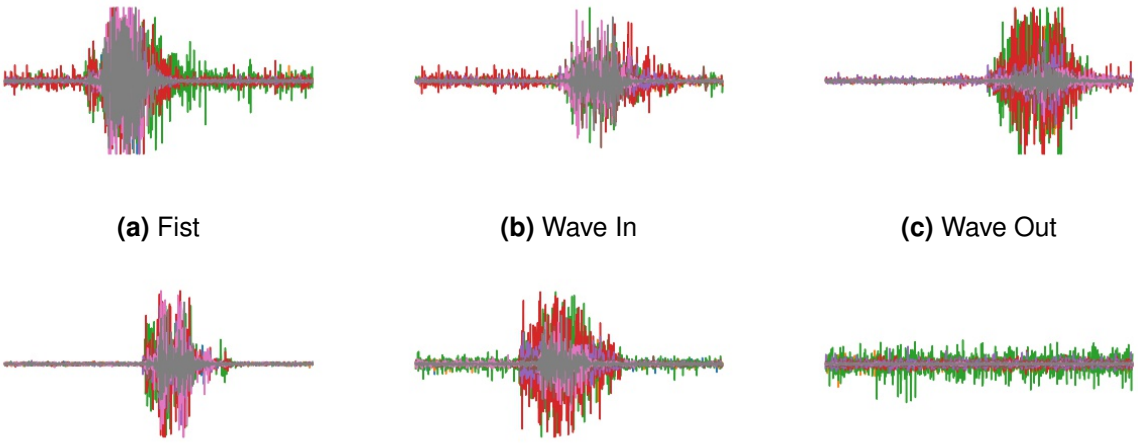


Figura 2.8: Grafica señales Raw EMG.
Fuente: autor

En la Figura 2.9 muestra las graficas de las señales mioeléctricas de los diferentes gestos a clasificar antes se ser preprocesadas.



(a) Fist **(b)** Wave In **(c)** Wave Out

(d) Double Tap **(e)** Fingers Spread **(f)** Relax

Figura 2.9: Gestos a clasificar sin preprocesamiento.
Fuente: autor

2.2.2 Preprocesamiento de Datos

El preprocesamiento de señales mioeléctricas emg implica la limpieza y la normalización de los datos antes de su análisis, es esencial para garantizar la precisión y la confiabilidad de los resultados del análisis, El proceso de preprocesamiento comienza con la rectificación de la señal, para luego filtrar utilizando un filtro pasa bajos ademas de normalizar la señal.

Normalización

Se utiliza para asegurar que los datos de los gestos a clasificar se encuentren en una escala comparable antes de su análisis.

Rectificación

La rectificación se realiza mediante la eliminación de los valores negativos de la señal, dejando solo los valores positivos. El proceso se lo realiza mediante un algoritmo de procesamiento de señales digital. La función abs(x) toma como entrada un número y devuelve su valor absoluto, es decir, elimina el signo negativo si existe

Filtrado

En el preprocesamiento de señales EMG, un filtro de paso bajo Butterworth Se aplica para eliminar las frecuencias de alta perturbación originadas por diferentes fuentes como el movimiento de la piel, el ruido del ambiente y otros factores. El filtro se ajusta para permitir que las frecuencias de interés, como las frecuencias de contracción muscular, pasen a través del filtro mientras que el ruido de alta frecuencia es atenuado.

Se requiere especificar el orden del filtro, la frecuencia de corte y las características de la señal para aplicar un filtro de paso bajo Butterworth en señales EMG. Una vez establecidos estos parámetros, se aplica el filtro a la señal utilizando un algoritmo de filtrado digital, como el filtro de Butterworth en python.

Para utilizar un filtro de paso bajo Butterworth en Python, se utiliza la biblioteca de procesamiento de señales "scipy". La biblioteca "scipy" proporciona la función "butter" para crear un filtro de Butterworth, y la función "filtfilt" para aplicar el filtro a una señal. La función "filtfilt" es una función de filtrado bidireccional que se utiliza para aplicar un filtro a una señal. A diferencia de la función "lfilter", que solo aplica el filtro en una dirección, "filtfilt" aplica el filtro en ambas direcciones, lo que permite

una mayor eliminación del ruido y una respuesta del filtro más suave. Es relevante considerar que este tipo de filtrado puede ser más costoso en términos de recursos debido a que requiere una mayor cantidad de memoria y tiempo de procesamiento.

Para el preprocesamiento de las señales EMG en este proyecto, se comienza con la rectificación de la señal y posteriormente se filtra mediante un filtro pasa bajo Butterworth de sexto orden con una frecuencia de corte de 20 Hz. Para determinar la frecuencia de corte adecuada, se separan los datos por canal y se ajustan los valores de diseño del filtro. En la Figura 2.10, se muestra el resultado del preprocesamiento para cada canal del gesto fist.

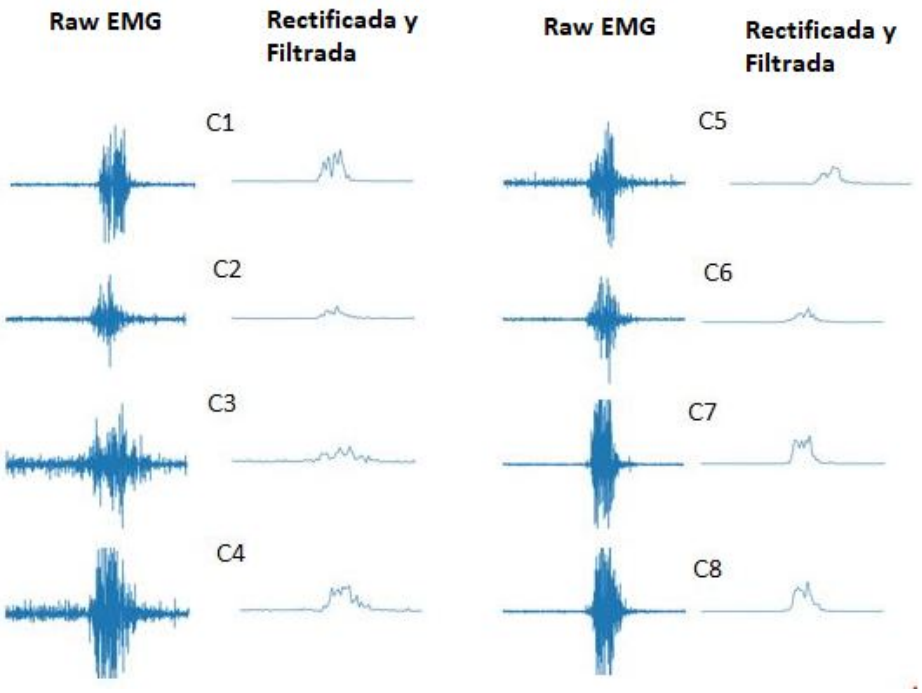
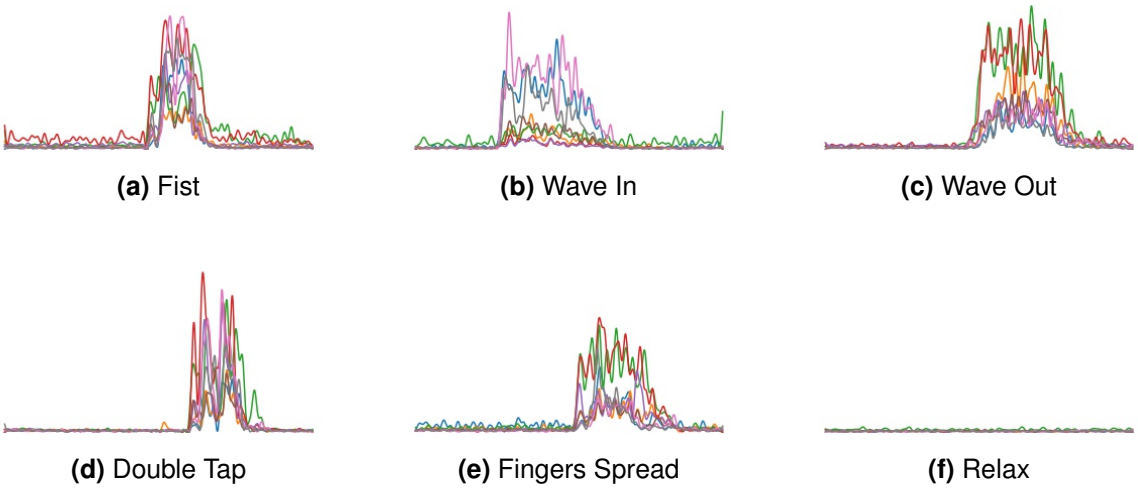


Figura 2.10: Señales preprocesadas por canal.
Fuente: autor

En la Figura 2.11 se puede apreciar el resultado de la rectificación y filtrado de cada gesto a clasificar.



(a) Fist **(b)** Wave In **(c)** Wave Out

(d) Double Tap **(e)** Fingers Spread **(f)** Relax

Figura 2.11: Señales de gestos a clasificar preprocesados.
Fuente: autor

2.2.3 Almacenamiento Datos

Las imágenes de los gestos de la mano obtenidas del dataset deben ser almacenadas en el entorno de google drive, previamente creando una carpeta llamada DatosGestos dentro de la cual nuevamente se crearan carpetas con los siguientes nombres, para poder identificar a que gesto corresponden: Double tap, fingersSpread, fist, relax, waveIn, waveOut como se muestra en el diagrama de la Figura 2.12.

Figura 2.12: Esquema general de la base de datos.
Fuente: autor

Las imágenes almacenadas en las carpetas de acuerdo al gesto a clasificar

cuentan con una dimensión de 432 x 288 pixeles que corresponden al ancho y alto respectivamente en formato jpg, se recopilan 3900 imágenes correspondiente a las muestras de 30 usuarios.

2.2.4 Creación del DataFrame

El esquema para crear un DataFrame y separarlo en conjuntos de entrenamiento y evaluación es el siguiente:

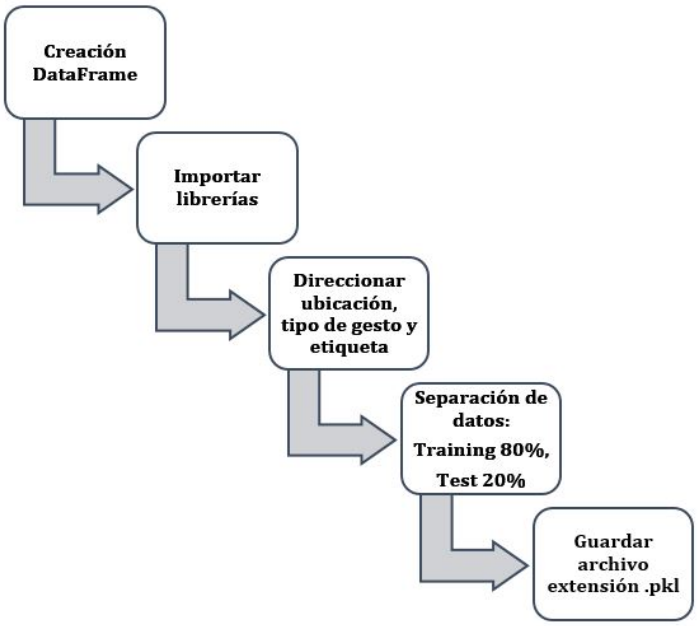


Figura 2.13: Proceso creación DataFrame.
Fuente: autor

Utilizando la librería pandas se puede generar un dataframe que contenga las 3900 imágenes almacenadas, con las columnas que indican la ubicación, el nombre de la señal y la etiqueta del gesto a clasificar. La cantidad de filas del dataframe se relaciona con la cantidad de imágenes. Esto se puede apreciar en la Figura 2.14.

	ubicacion	nom_señal	etiqueta
0	/content/drive/MyDrive/Colab Notebooks/DatosGe...	img_0.jpg	relax
1	/content/drive/MyDrive/Colab Notebooks/DatosGe...	img_1.jpg	relax
2	/content/drive/MyDrive/Colab Notebooks/DatosGe...	img_2.jpg	relax
3	/content/drive/MyDrive/Colab Notebooks/DatosGe...	img_3.jpg	relax
4	/content/drive/MyDrive/Colab Notebooks/DatosGe...	img_4.jpg	relax
...	...	...	...
3895	/content/drive/MyDrive/Colab Notebooks/DatosGe...	img_3870.jpg	fingersSpread
3896	/content/drive/MyDrive/Colab Notebooks/DatosGe...	img_3871.jpg	fingersSpread
3897	/content/drive/MyDrive/Colab Notebooks/DatosGe...	img_3872.jpg	fingersSpread
3898	/content/drive/MyDrive/Colab Notebooks/DatosGe...	img_3873.jpg	fingersSpread
3899	/content/drive/MyDrive/Colab Notebooks/DatosGe...	img_3874.jpg	fingersSpread

3900 rows × 3 columns

Figura 2.14: DataFrame de gestos.
Fuente: autor

La Figura 2.15 presenta la distribución de la cantidad de imágenes por gesto en el dataframe creado, en el cual cada gesto tiene 750 imágenes, excepto el gesto de relajación que cuenta con 150 imágenes

```
[ ]  df.etiqueta.value_counts()

waveOut          750
doubleTap        750
fist             750
waveIn           750
fingersSpread    750
relax            150
Name: etiqueta, dtype: int64
```

Figura 2.15: Cantidad de gestos.
Fuente: autor

Podemos dividir el dataframe en subconjuntos de entrenamiento y prueba, comúnmente el subconjunto de entrenamiento se compone del 80 % del conjunto total, mientras que el 20 % restante se utiliza para formar el subconjunto de prueba.

La selección de los datos se realiza de manera aleatoria para garantizar que los conjuntos sean representativos del conjunto de datos completo y eliminar cualquier sesgo de selección, lo que puede reducir los efectos de las diferencias entre los datos y determinar las características más importantes. El proceso de separación del dataframe se lleva a cabo mediante la utilización del código que se muestra en la Figura 2.16, en el cual se especifican los parámetros necesarios.

```
from sklearn.model_selection import train_test_split

[ ] df_train,df_test=train_test_split(df,test_size=0.2,random_state=42)
```

Figura 2.16: Código separar dataframe.
Fuente: autor

lo que se obtiene es un subconjunto de datos de entrenamiento llamado df_train, con datos aleatorios como muestra la Figura 2.17 y un df_test con el resto de datos.

```
df_train
```

	ubicacion	nom_señal	etiqueta
247	drive/MyDrive/Colab Notebooks/DatosGestos/wave...	img_442.jpg	waveOut
2591	drive/MyDrive/Colab Notebooks/DatosGestos/wave...	img_931.jpg	waveIn
1822	drive/MyDrive/Colab Notebooks/DatosGestos/fist...	img_857.jpg	fist
3483	drive/MyDrive/Colab Notebooks/DatosGestos/fing...	img_1778.jpg	fingersSpread
2771	drive/MyDrive/Colab Notebooks/DatosGestos/wave...	img_1846.jpg	waveIn
...	...	...	...
1130	drive/MyDrive/Colab Notebooks/DatosGestos/doub...	img_1280.jpg	doubleTap
1294	drive/MyDrive/Colab Notebooks/DatosGestos/doub...	img_2074.jpg	doubleTap
860	drive/MyDrive/Colab Notebooks/DatosGestos/wave...	img_3680.jpg	waveOut
3507	drive/MyDrive/Colab Notebooks/DatosGestos/fing...	img_1907.jpg	fingersSpread
3174	drive/MyDrive/Colab Notebooks/DatosGestos/fing...	img_104.jpg	fingersSpread

3120 rows × 3 columns

Figura 2.17: Subconjunto entrenamiento.
Fuente: autor

La cantidad de datos de los subconjuntos de entrenamiento y pruebas podemos observar como muestra en la Figura 2.18 de acuerdo a cada gesto a clasificar y a la proporción establecida.

```
[ ] df_train.etiqueta.value_counts()

fingersSpread    610
waveIn           607
fist             605
doubleTap        592
waveOut          589
relax            117
Name: etiqueta, dtype: int64
```

(a) Datos entrenamiento.

```
df_test.etiqueta.value_counts()

waveOut          161
doubleTap        158
fist             145
waveIn           143
fingersSpread    140
relax             33
Name: etiqueta, dtype: int64
```

(b) Datos pruebas.

Figura 2.18: Dataframe Gestos.
Fuente: autor

Para guardar el dataframe en Google Drive, se utiliza el módulo pickle que utiliza protocolos binarios para convertir una estructura de objetos de Python en una secuencia de bytes. El proceso de convertir una jerarquía de objetos de Python en una secuencia de bytes se llama "pickling", mientras que la operación inversa de convertir una secuencia de bytes en una jerarquía de objetos se llama ünpickling"(Python.org, 2022), esto crea un archivo de extensión pickle, con el nombre: base_de_datos_Gestos30User.pkl

2.3 DISEÑO DE RED

2.3.1 Selección modelo de machine learning

Los progresos en los sistemas de aprendizaje automático, tales como las redes neuronales artificiales (ANN), máquinas de vectores de soporte (SVM) y en la actualidad las redes neuronales convolucionales (CNN), han permitido el desarrollo de nuevos sistemas inteligentes con una precisión casi equiparable a la humana. La importancia de estos sistemas pueden agregar una dimensión de movilidad para conjuntos robóticos y la liberación de tareas monótonas para el ser humano como en el caso del reconocimiento de caracteres, entre otras de interés comercial como Big Data (Smola & Vishwanathan, 2008).

Se han empleado diversos modelos de machine learning para clasificar gestos de la mano. Con el objetivo de seleccionar el modelo adecuado para este proyecto, se llevó a cabo un análisis que se presenta en la Tabla 2.1.

Tipo Machine Learning	SVM	Redes Neuronales	CNN
Entrenamiento	Supervisado y no supervisado	Supervisado y no supervisado	Supervisado y no supervisado
Modelo de funcionamiento	Hiperplanos	Producto, interconexión y funciones de activación	Similar a la corteza visual de seres vivos
Aplicaciones	Clasificación y Regresión	Clasificación, predicción, optimización, Control	Clasificación y reconocimiento de imágenes, Procesamiento de voz
Tasa de Error en Dataset MINST	0.88 %	2.8 %	0.61 %

Tabla 2.1: Comparativa entre modelos de machine learning
Fuente: Autor

La comparativa anterior muestra diversos modelos de aprendizaje de máquina, y se observa que las Redes Neuronales Convolucionales son las que presentan mejores resultados. Actualmente, las CNN han evolucionado y permiten a los desarrolladores crear algoritmos con una alta precisión.

Una estrategia para utilizar CNN es emplear redes pre-entrenadas con bases de datos grandes y ajustarlas para resolver el problema específico de interés. En la Figura 2.19 se muestra la estructura de una CNN aplicada a datos EMG.

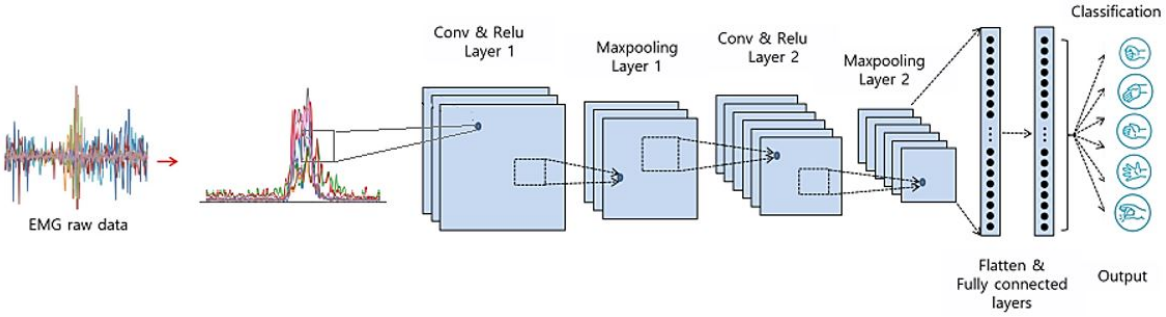


Figura 2.19: Estructura de una CNN con datos de imagen EMG
Fuente:autor

2.3.2 Transfer Learning

El aprendizaje de transferencia se refiere a la técnica de utilizar un modelo previamente entrenado con un gran conjunto de datos, generalmente de ImageNet en el campo de la visión artificial, y adaptarlo al nuevo conjunto de datos en cuestión. La idea de utilizar este método es que ha aprendido a reconocer varias características o patrones en todos estos datos y que nuestros datos se beneficiará de este conocimiento, especialmente si el conjunto de datos es pequeño, en comparación con iniciar con un modelo desde cero. Se ha demostrado en una gran variedad de tareas que la transferencia de aprendizaje casi siempre da mejores resultados.

La estructura básica del transfer learning en aprendizaje automático consiste en tres pasos:

- Selección de un modelo previamente entrenado: Se elige un modelo previamente entrenado en una tarea que sea similar a la tarea en la que se desea aplicar el transfer learning.

- Reutilización de las capas intermedias del modelo: La idea detrás de la reutilización de las capas intermedias es que estas capas contienen características genéricas que pueden ser reutilizadas en la nueva tarea.

- Entrenamiento de la última capa: La última capa del modelo es entrenada con los datos de la nueva tarea para que pueda adaptarse a la nueva tarea.

2.3.3 Selección modelo de entrenamiento

La arquitectura de la CNN se fundamenta en la estructura y el funcionamiento de la corteza visual, con el objetivo de simular el modelo de interconexión de las neuronas presentes en el cerebro. La CNN incluye tres tipos de capas que son convolucionales, de agrupación y totalmente conectadas. En la CNN, a medida que las capas se multiplican, la red se vuelve más difícil de entrenar y también

la precisión alcanza la saturación y luego comienza a disminuir. (Ozdemir y col., 2020)

Para este caso se consideraron los modelos de Alexnet, VGG, Resnet, DenseNet para seleccionar la arquitectura de la red.

Nombre del modelo	**Número de parámetros [Millones]**	**ImageNet Top 1 Precisión**
AlexNet	60M	63,3 %
VGG 19	144 M	74,5 %
ResNet-50	26 M	77,15 %
ResNet-152	60 M	78,57 %
DenseNet-121	8 M	74,98 %

Tabla 2.2: Comparativa arquitectura
Fuente: (Adaloglou, 2021)

En la Tabla 2.2 muestra una comparativa de las arquitecturas mas utilizadas, ademas se toma en cuenta que el modelo ResNet (red neuronal residual) se convierte en el ganador del ILSVRC, ImageNet Large Scale Visual Recognition Challenge 2015 en clasificación, detección y localización de imágenes, así como en el Ganador de MS COCO 2015 en detección y segmentación. (He y col., 2015)

La Figura 2.20 presenta algunas arquitecturas de red para ImageNet, incluyendo el modelo VGG-19 a la izquierda, una red simple con 34 capas de parámetros en el centro, y una red residual con 34 capas de parámetros a la derecha. En esta última, los atajos de puntos se utilizan para aumentar las dimensiones.

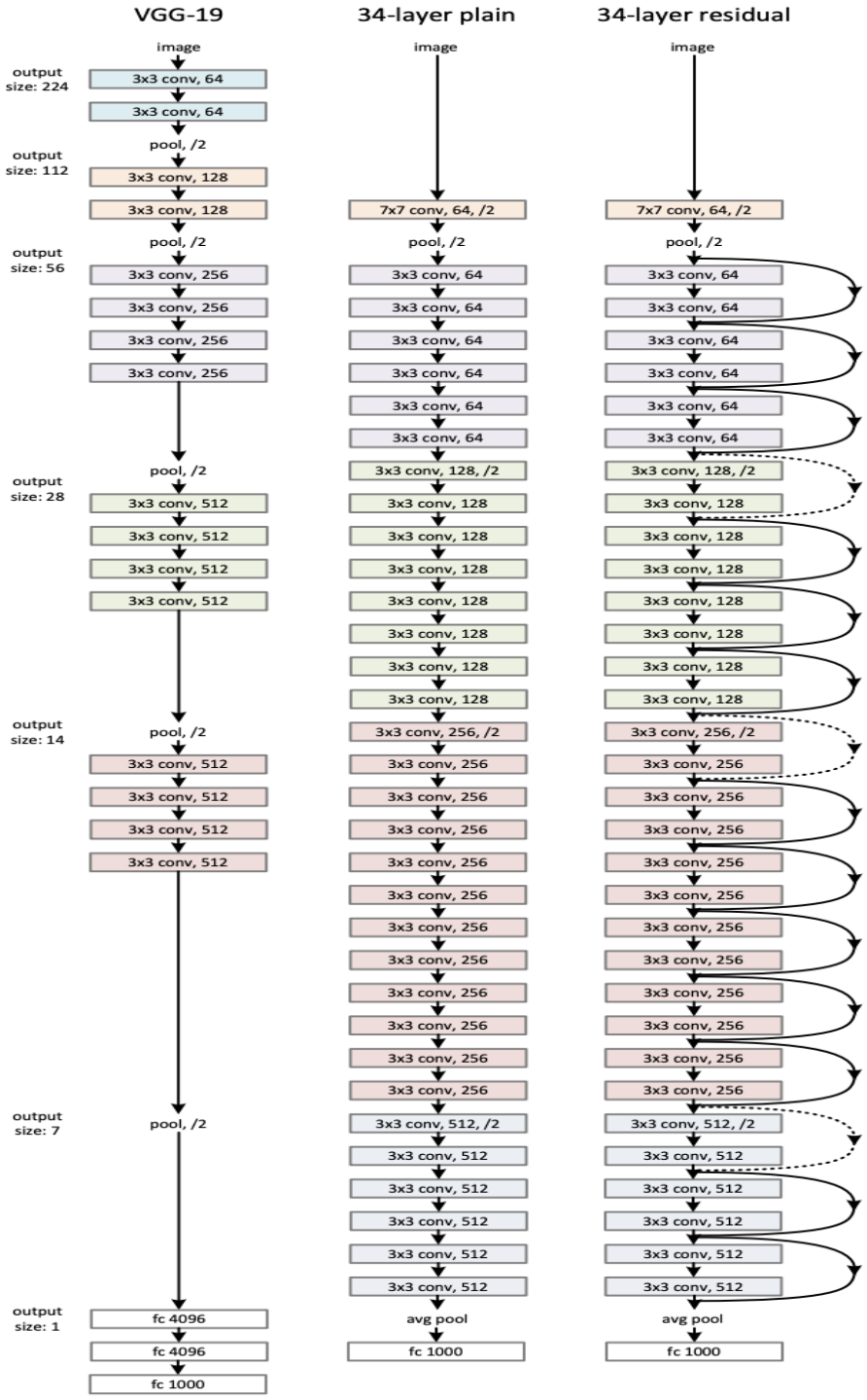


Figura 2.20: Ejemplos arquitecturas de red por imageNet
Fuente: (He y col., 2015)

De la Figura 2.20 La red simple de 34 capas (medio) se trata como la red más profunda de VGG-19 , es decir, más capas conv. La red residual de 34 capas (ResNet) (derecha) es la simple con la adición de una conexión de salto/acceso directo.

En este trabajo, en base al análisis realizado se utilizó el modelo ResNet para entrenar imágenes de gestos creadas a partir de señales mioeléctricas registradas durante los gestos de la mano.

2.3.4 Arquitectura de red ResNet

Una Red Neuronal Residual (ResNet) que es una subclase de CNN, es un modelo de red neuronal artificial que se utiliza en tareas de aprendizaje automático, como la clasificación, detección de objetos y segmentación de imágenes. La arquitectura de ResNet se distingue por sus bloques residuales, que permiten que la información fluya a través de múltiples capas de la red sin pérdida de información, como se ilustra en la Figura 2.21. La arquitectura de ResNet consta de múltiples bloques de identidad concatenados, cada uno de los cuales se compone de capas totalmente conectadas, capas de normalización, capas de activación y capas de salida.

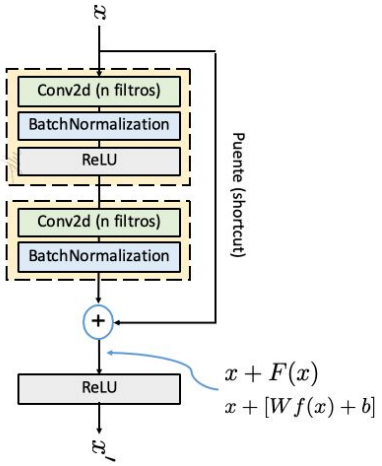


Figura 2.21: Aprendizaje residual: un bloque de construcción
Fuente: (He y col., 2015)

El aprendizaje residual ayuda a desentrañar ese problema de reducción de la precisión. El aprendizaje residual utiliza enlaces de la forma más rápida como una técnica de entrenamiento para vincular directamente la entrada no solo a la siguiente contigua sino también a otras capas siguientes, para el entrenamiento de la red neuronal. En la Figura 2.22, hay un resumen del tamaño de salida en cada capa y la dimensión de los núcleos convolucionales en cada punto de la estructura.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figura 2.22: Tamaños de salidas y núcleos convolucionales para ResNet
Fuente: (He y col., 2015)

La Figura 2.23 proporciona una interpretación visual de cómo las capas se van profundizando en la arquitectura ResNet, basada en la Figura 2.22.

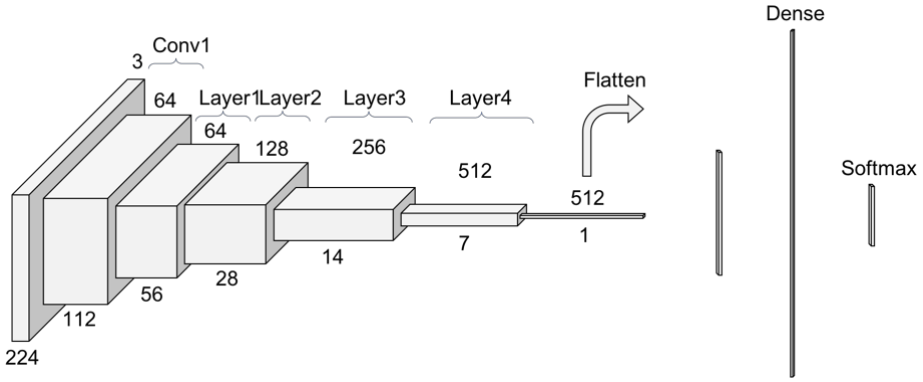


Figura 2.23: Explicación gráfica ResNet 34
Fuente: (He y col., 2015)

La CNN que se emplea se desempeña como un clasificador, tomando como entrada la imagen EMG preprocesada y procesándola a través de varias capas para obtener la clasificación del gesto. En la Figura 2.24, se muestra la estructura de una CNN ResNet de 34 capas que se usa para clasificar gestos utilizando señales EMG. Es importante señalar que la estructura varía según los datos de la Figura 2.22 para las diferentes variantes del modelo de red.

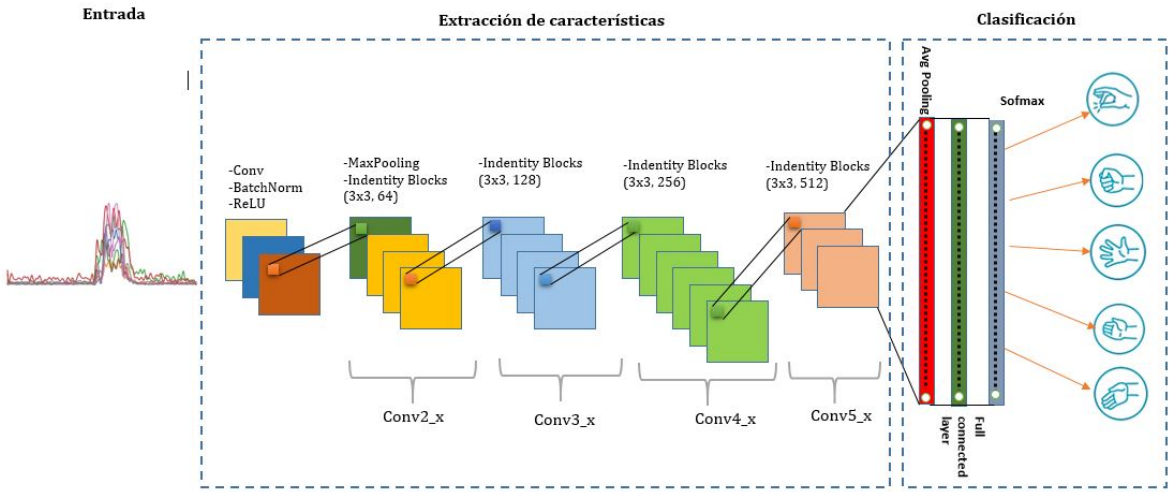


Figura 2.24: Estructura ResNet34 con datos EMG
Fuente: autor

En el contexto de la clasificación de señales mioeléctricas EMG, el funcionamiento de una ResNet se puede interpretar de la siguiente manera:

Extracción de características

La ResNet se utiliza para aprender representaciones complejas del gesto o la señales EMG mediante la extracción relevantes de sus características. Estas pueden incluir patrones temporal y espacial, así como la información de frecuencia de la señal EMG.

Identificación de patrones

La ResNet es capaz de identificar patrones en las señales EMG que son relevantes para la clasificación. Conforme la información fluye a través de múltiples capas de la red, la ResNet es capaz de aprender representaciones cada vez más complejas de los datos.

Clasificación

Una vez entrenada, la ResNet se utiliza para clasificar nuevas señales EMG. La red recibe la señal como entrada y, luego de procesarla a través de sus capas, emite una predicción de la clase a la que corresponde la señal de entrada..

Solución al problema de vanishing gradients

La ResNet tiene una arquitectura diseñada para solucionar el problema de los gradientes desvanecidos, que pueden ocurrir en redes profundas, permitiendo una mejor capacidad de aprendizaje. El fenómeno del vanishing gradients es un problema importante en redes profundas, ya que impide que la red pueda aprender representaciones complejas de los datos. Al no poder actualizar adecuadamente sus pesos, las capas internas no pueden aprender patrones relevantes en los datos, lo que afecta la precisión y rendimiento.

2.4 ENTRENAMIENTO DE RED

A continuación se presenta el procedimiento para entrenar la red neuronal convolucional (CNN) para clasificar los gestos:

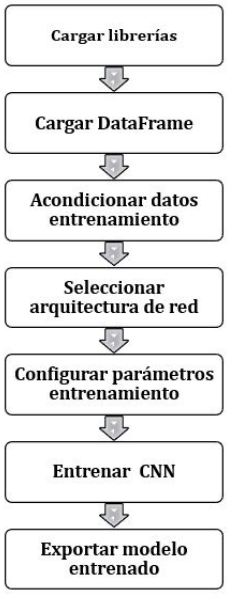


Figura 2.25: Proceso de entrenamiento CNN.
Fuente: autor

Para iniciar con el entrenamiento se debe verificar que el entorno de ejecución en google colab se encuentre seleccionado la GPU, y revisar cuales son las características de RAM, tarjeta grafica asignadas para la ejecución del código con el comando de la Figura 2.26 se puede obtener la información necesaria.

```
[ ] !nvidia-smi

Wed May  4 17:28:09 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   30C    P8    28W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Figura 2.26: Características GPU.
Fuente: autor

Se deben importar las librerías de aprendizaje profundo necesarias en nuestro caso se hará uso de Fastai, es una librería de aprendizaje automático de código

abierto que se basa en PyTorch y proporciona una interfaz sencilla para el desarrollo de modelos de aprendizaje automático. Con Fastai, es posible construir una arquitectura ResNet con un número reducido de líneas de código.

Después, es necesario cargar el conjunto de datos previamente creado llamado base_de_datos_Gestos30User.pkl, que se ubica en Google Drive, utilizando los comandos que se presentan en la Figura 2.27.

```
cargar_df=os.path.join(path,"/content/drive/MyDrive/Colab Notebooks/base_de_datos_Gestos30User.pkl")
print(cargar_df)
```

```
/content/drive/MyDrive/Colab Notebooks/base_de_datos_Gestos30User.pkl
```

```
[ ] df_train,df_test=pd.read_pickle(cargar_df)
```

```
[ ] df_train['set']='train'
    df_test['set']='test'
```

```
[ ] df=pd.concat([df_train,df_test],ignore_index=True,sort=False)
    df
```

Figura 2.27: Código cargar DF.
Fuente: autor

El marco de datos final, que contiene los datos de entrenamiento y prueba de los gestos con sus etiquetas y ubicaciones correspondientes, se muestra en la Figura 2.28, como se mencionó anteriormente en secciones anteriores.

	ubicacion	nom_señal	etiqueta	set
0	drive/MyDrive/Colab Notebooks/DatosGestos/wave...	img_442.jpg	waveOut	train
1	drive/MyDrive/Colab Notebooks/DatosGestos/wave...	img_931.jpg	waveIn	train
2	drive/MyDrive/Colab Notebooks/DatosGestos/fist...	img_857.jpg	fist	train
3	drive/MyDrive/Colab Notebooks/DatosGestos/fing...	img_1778.jpg	fingersSpread	train
4	drive/MyDrive/Colab Notebooks/DatosGestos/wave...	img_1846.jpg	waveIn	train
...	...	...	...	...
3895	drive/MyDrive/Colab Notebooks/DatosGestos/doub...	img_2593.jpg	doubleTap	test
3896	drive/MyDrive/Colab Notebooks/DatosGestos/fing...	img_3201.jpg	fingersSpread	test
3897	drive/MyDrive/Colab Notebooks/DatosGestos/wave...	img_1456.jpg	waveIn	test
3898	drive/MyDrive/Colab Notebooks/DatosGestos/fist...	img_3834.jpg	fist	test
3899	drive/MyDrive/Colab Notebooks/DatosGestos/wave...	img_3415.jpg	waveOut	test

3900 rows × 4 columns

Figura 2.28: Data Frame entrenamiento.
Fuente: autor

Antes de que la base de datos pueda ser utilizada en el modelo de entrenamiento,

es necesario ajustar o acondicionar los datos según las instrucciones en la Figura 2.29.

```
[ ]  tecnicas=get_transforms(flip_vert=False,max_rotate=0,max_lighting=None)

[ ]  data = (ImageList.from_df(df, path=path, cols='ubicacion')
            .split_by_idxs(train_index, test_index)
            .label_from_df(cols='etiqueta')
            .transform(tecnicas, size=224)
            .databunch(bs=4)
            .normalize(imagenet_stats))
```

Figura 2.29: Acondicionamiento de datos.
Fuente: autor

Configuramos los parámetros de datos de acuerdo la Tabla 2.3

Parámetro	Definición
ImageList.from_df(path)	Ubicación de los datos
.split_by_idxs()	Dividir en train/test
.label_from_df()	Etiqueta en el nombre de archivo de los datos
transform(tecnicas, size=224)	Aumento de datos, tamaño img de 224
.databunch(bs=6)	Tamaño de lote
normalize(imagenet_stats)	Normalizar datos

Tabla 2.3: Parámetros de función
Fuente: autor

Aquí todas las imágenes están normalizadas a un tamaño de 224 por 224 píxeles que es la resolución admitida por el modelo resnet. La "bs" significa tamaño de lote, que es el conjunto de imágenes que procesa el modelo al mismo tiempo. Se puede proporcionar una explicación más detallada de la función "normalize", que consiste en ajustar los datos de manera que tengan una media de cero y una desviación estándar de uno. Esto se hace para que las redes neuronales se sientan más cómodas trabajando con datos que sigan este patrón, ya que de esta manera, los datos están más estandarizados y son más fáciles de comparar y manipular, esta técnica consiste en restar la media de la variable a cada puntuación y dividirla por la desviación estándar de esa variable. Al emplear el modelo preentrenado de imagenet, es necesario emplear la media y la desviación estándar correspondientes a los datos de imagenet originales en lugar de las del conjunto de datos actual. Por

lo tanto, se almacenan los valores de media y desviación estándar de imagenet en la variable ïmagenet_stats".

Para iniciar el proceso de entrenamiento del modelo con los datos preparados y el modelo configurado, se utiliza el comando çnn_learner", el cual se muestra en la Figura 2.30.

Figura 2.30: Entrenamiento de red.
Fuente: autor

Los argumentos del comando para el entrenamiento incluyen la base de datos y el modelo de red seleccionado (en este caso resnet34), así como la obtención de varias métricas como exactitud, tasa de error. Además, se configura el número de épocas mediante el comando ANN.fit_one_cycle. Cabe destacar que una época se refiere a un ciclo completo de entrenamiento en todo el conjunto de datos.

Se puede entrenar la red neuronal convolucional con un número limitado de épocas debido a las especificaciones previamente establecidas. El número de épocas se elige incrementándolo gradualmente hasta que la métrica de exactitud con los datos de validación comienza a disminuir, incluso si la exactitud con los datos de entrenamiento sigue aumentando. Si se detecta una posible situación de sobreajuste o overfitting, es necesario detener el entrenamiento.

Una vez finalizado el entrenamiento se pueden observar los valores de exactitud y pérdidas, que también pueden ser graficados para un posterior predicción o clasificación.

Para realizar posteriores clasificaciones de gestos de la mano utilizando modelo de red neuronal convolucional desarrollado se pueden seguir los siguientes pasos.

Exportar el modelo entrenado

Una vez entrenado el modelo, se puede utilizar la función learn.export para guardar el modelo en un archivo .pkl, que puede ser cargado posteriormente con la función load_learner.

Cargar el modelo

Para cargar el modelo, se puede utilizar la función load_learner y proporcionar el path del archivo .pkl y el path de la carpeta donde se encuentran los datos. La función devolverá un objeto Learner que contiene el modelo entrenado.

Realizar clasificaciones o predicciones

Se pueden utilizar los métodos predict o get_preds del objeto Learner. Estos métodos toman una imagen o una lista de imágenes y devuelven las predicciones del modelo.

Un ejemplo de los comandos necesarios se puede apreciar a continuación:

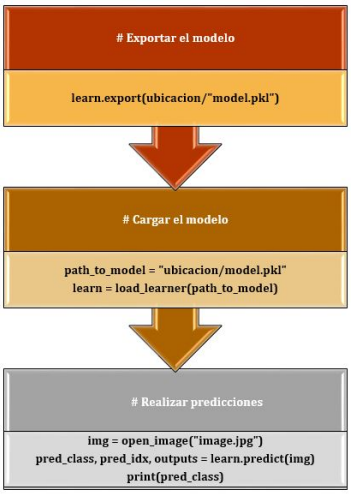


Figura 2.31: Comandos para utilizar red entrenada.
Fuente: autor

El diagrama de flujo para la clasificación de nuevos gestos con datos EMG en Colab se presenta en la Figura 2.32.

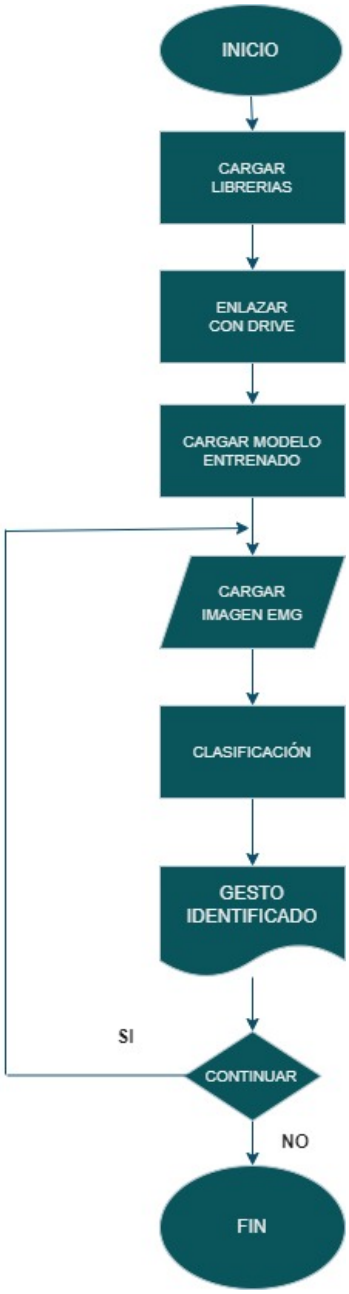


Figura 2.32: Diagrama de flujo algoritmo de clasificación gestos EMG.
Fuente: autor

3 RESULTADOS Y DISCUSIÓN

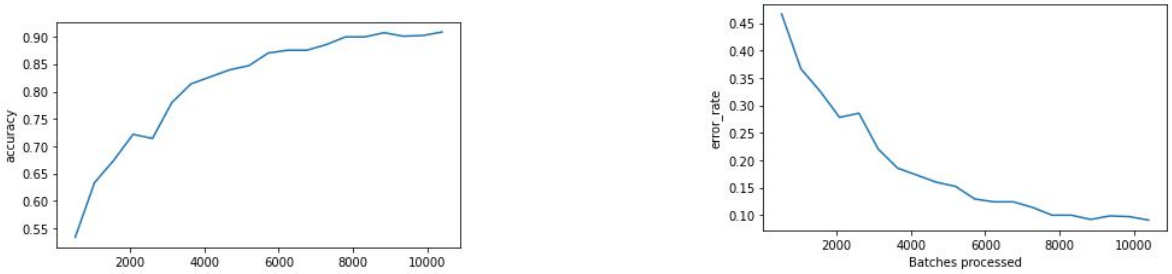
En esta sección se presentan los resultados obtenidos durante las pruebas de la red neuronal convolucional ResNet para la clasificación de gestos de mano mediante señales mioeléctricas. Se evaluaron diferentes parámetros, tales como el tamaño de lote, número de épocas de entrenamiento y el número de capas del modelo ResNet. Además, se realizaron pruebas utilizando datos sin preprocesar. A continuación, se describen los resultados más importantes obtenidos en las diferentes pruebas y entrenamientos, incluyendo las métricas de clasificación de gestos, así como la matriz de confusión obtenida.

3.1 PRUEBAS CON DATOS PREPROCESADOS

En esta sección se mostrarán los resultados obtenidos en las diferentes variantes del modelo, comenzando con el modelo ResNet34 y aumentando gradualmente el número de capas hasta llegar al modelo ResNet152. Todos los entrenamientos se realizaron con los mismos parámetros, con un tamaño de lote de 6 y 20 épocas de entrenamiento.

3.1.1 Entrenamiento con modelo ResNet34

Los resultados del modelo de red Resnet34 se presentan a continuación. La Figura 3.1 muestra la gráfica del accuracy o exactitud vs batches processed o lotes procesados junto con la tasa de error durante las 20 épocas de entrenamiento, se puede apreciar que los valores de accuracy aumentan sin mayores variaciones mientras el entrenamiento avanza.



(a) Exactitud modelo ResNet34

(b) Tasa de error modelo ResNet34

Figura 3.1: Entrenamiento Resnet34

La gráfica de la Figura 3.2 muestra cómo los valores de pérdida disminuyen a medida que avanza el entrenamiento.

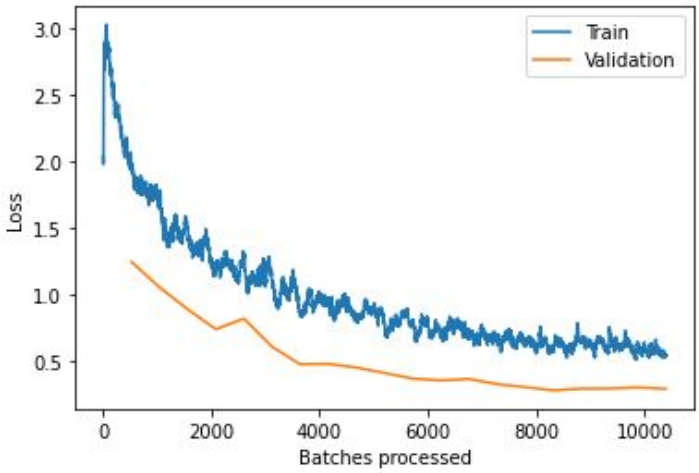


Figura 3.2: Pérdidas del modelo ResNet34

Los resultados obtenidos del entrenamiento utilizando el modelo resnet34 se presentan en la Tabla 3.1.

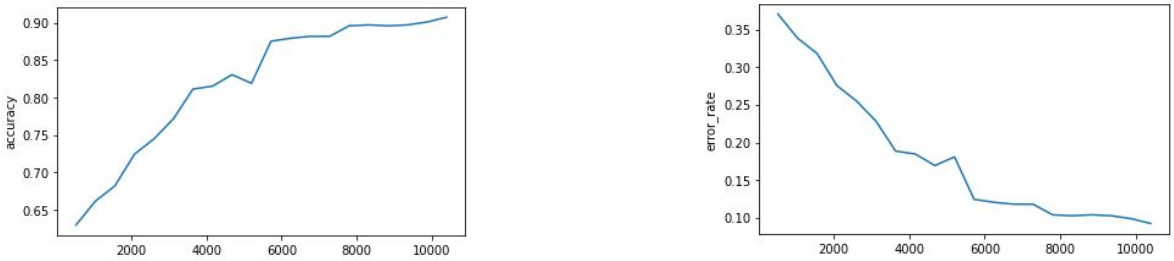
Épocas	Perdidas de entrenamiento	Perdidas de validación	Exactitud	Tasa de error
0	1.97	1.25	0.53	0.47
4	1.29	0.82	0.71	0.29
9	0.79	0.41	0.85	0.15
14	0.66	0.31	0.90	0.10
19	0.55	0.30	0.91	0.09

Tabla 3.1: Resultados de entrenamiento ResNet34
Fuente: autor

Los resultados reflejan que el modelo es entrenado de forma adecuada con una exactitud del 91 %, y una tasa de error del 9 %,

3.1.2 Entrenamiento con modelo ResNet50

El historial de entrenamiento tanto de exactitud, tasa de error se puede observar en la Figura 3.3. de la misma manera la red neuronal aprende de forma continua con pocas oscilaciones.



(a) Exactitud modelo ResNet50 **(b)** Tasa de error modelo ResNet50

Figura 3.3: Entrenamiento Resnet50

De forma similar, en la gráfica de la Figura 3.4 se puede observar una disminución en los valores de pérdida a medida que avanzan las épocas de entrenamiento.

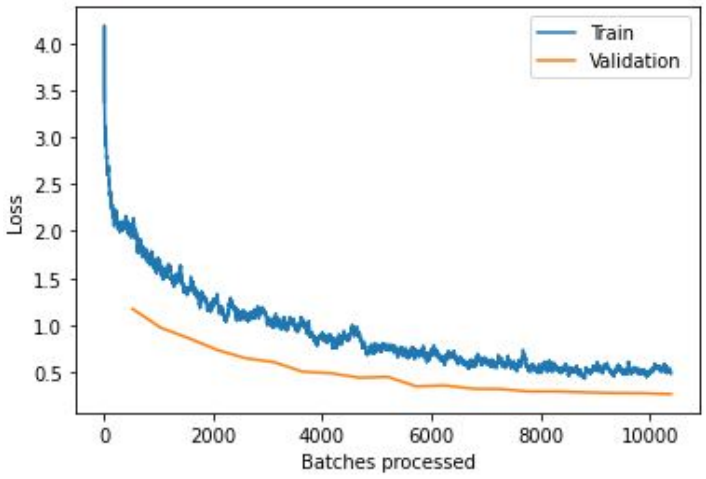


Figura 3.4: Pérdidas del modelo ResNet50

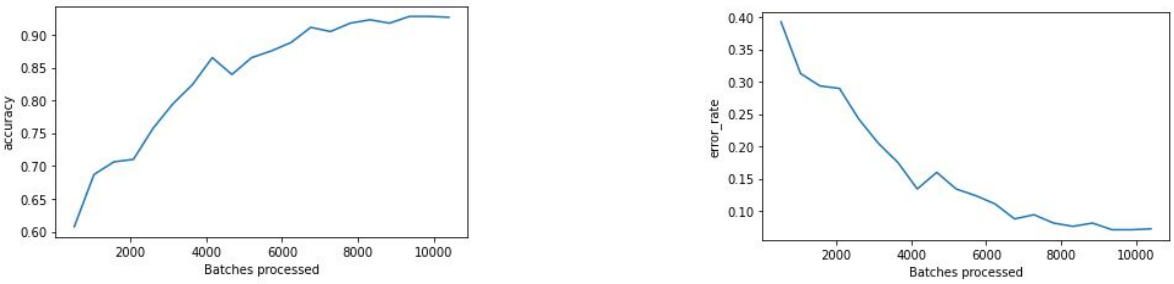
Épocas	Perdidas de entrenamiento	Perdidas de validación	Exactitud	Tasa de error
0	1.93	1.17	0.63	0.37
4	1.08	0.65	0.74	0.26
9	0.77	0.45	0.82	0.18
14	0.53	0.29	0.90	0.10
19	0.48	0.27	0.91	0.09

Tabla 3.2: Resultados de entrenamiento ResNet50
Fuente: autor

De la Tabla 3.2 se puede observar que se tiene una exactitud del 91 %, con una tasa de error de 9 %, se puede apreciar que se obtienen los mismos valores del exactitud y tasa de error del modelo anterior pero disminuyen los valores de perdidas de entrenamiento y validación.

3.1.3 Entrenamiento con modelo ResNet101

Los resultados del entrenamiento utilizando el modelo ResNet101 se presentan en esta sección. La figura 3.5 representa la gráfica de la exactitud frente a los lotes procesados y la tasa de error durante las 20 épocas de entrenamiento.



(a) Exactitud modelo ResNet101 **(b)** Tasa de error modelo ResNet101

Figura 3.5: Entrenamiento Resnet101

En la Figura 3.5 se puede observar que se tiene una buena exactitud pero se tiene ciertas oscilaciones durante las épocas de entrenamiento.

Similarmente, en la gráfica de la Figura 3.6 se evidencia que los valores de pérdida disminuyen a medida que transcurren las épocas de entrenamiento.

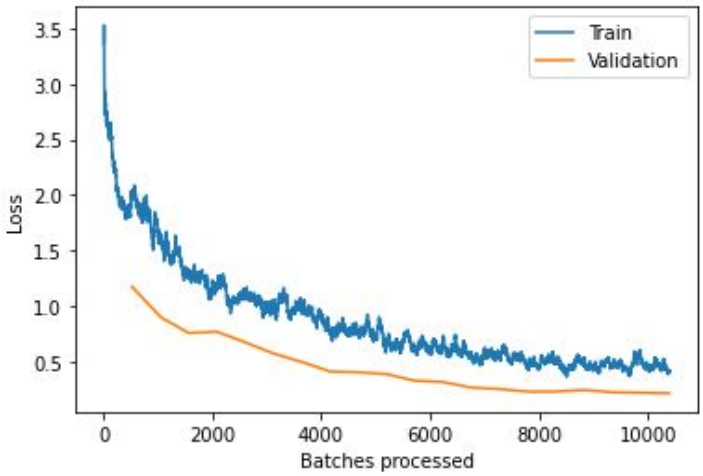


Figura 3.6: Pérdidas del modelo ResNet101
Fuente: autor

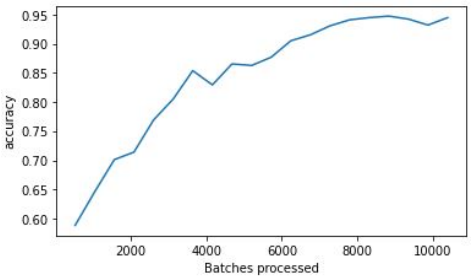
Épocas	Perdidas de entrenamiento	Perdidas de validación	Exactitud	Tasa de error
0	1.98	1.17	0.61	0.39
4	1.08	0.67	0.76	0.24
9	0.63	0.39	0.87	0.13
14	0.48	0.23	0.92	0.08
19	0.42	0.21	0.93	0.07

Tabla 3.3: Resultados de entrenamiento ResNet101
Fuente: autor

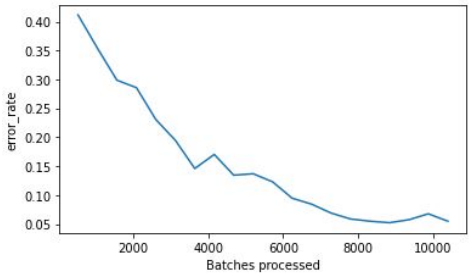
De la Tabla 3.3 se puede observar que se tiene una exactitud del 93 %, con una tasa de error de 7 % ademas de disminuir las perdidas de entrenamiento y validación, mejorando los resultados del modelo con resnet50.

3.1.4 Entrenamiento con modelo ResNet152

Por último, se exponen los resultados del entrenamiento realizado utilizando el modelo de red resnet152. En la Figura 3.7 se puede observar la gráfica que muestra la exactitud en función del número de lotes procesados y la tasa de error durante las 20 épocas de entrenamiento.



(a) Exactitud modelo ResNet152



(b) Tasa de error modelo ResNet152

Figura 3.7: Entrenamiento Resnet152

En la Figura 3.7 se puede observar que se tiene una buena exactitud a medida que avanza el entrenamiento. ademas que en la Figura 3.8 la pérdida disminuye a medida que avanza el entrenamiento, de manera similar a lo observado en los gráficos anteriores del modelo resnet.

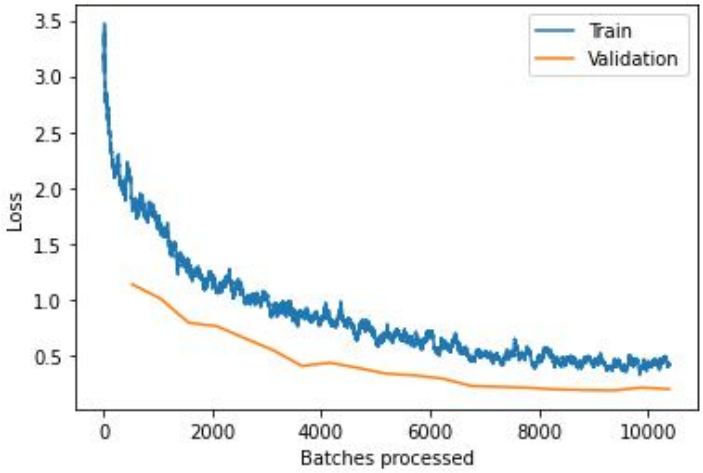


Figura 3.8: Pérdidas del modelo ResNet152

Épocas	Perdidas de entrenamiento	Perdidas de validación	Exactitud	Tasa de error
0	1.82	1.13	0.59	0.41
4	0.95	0.65	0.77	0.23
9	0.68	0.33	0.86	0.14
14	0.55	0.21	0.94	0.06
19	0.41	0.20	0.94	0.06

Tabla 3.4: Resultados de entrenamiento ResNet152
Fuente: autor

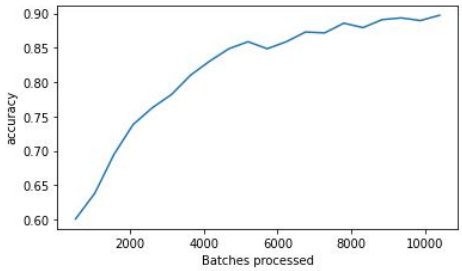
De la Tabla 3.4 se puede observar que se tiene una exactitud del 94 %, con una tasa de error de 6 %, mejorando un mínimo los resultados con el modelo resnet101.

3.2 PRUEBAS CON DATOS SIN PREPROCESAR

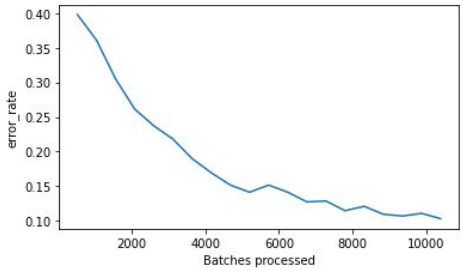
Se presentarán los resultados de entrenamiento obtenidos con los datos sin preprocesar con las cuatro variantes del modelo ResNet, todos establecidos con los mismos parámetros que fueron utilizados con los datos preprocesados

3.2.1 Entrenamiento con modelo ResNet34

Los resultados del modelo de red Resnet34 sin preprocesamiento de datos se presentan a continuación. La Figura 3.9 muestra la gráfica del accuracy o exactitud vs batches processed o lotes procesados junto con la tasa de error durante las 20 épocas de entrenamiento, se puede apreciar que los valores aumentan sin mayores variaciones mientras el entrenamiento avanza.



(a) Exactitud modelo ResNet34



(b) Tasa de error modelo ResNet34

Figura 3.9: Entrenamiento Resnet34

Los valores de pérdida disminuyen a medida que avanzan las épocas de entrenamiento, según se puede apreciar en la Figura 3.10.

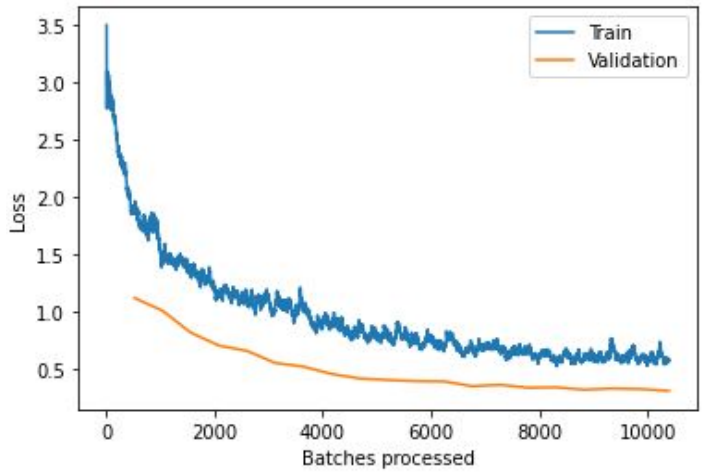


Figura 3.10: Pérdidas del modelo ResNet34

Los resultados obtenidos del entrenamiento utilizando el modelo resnet34 se presentan en la Tabla 3.5.

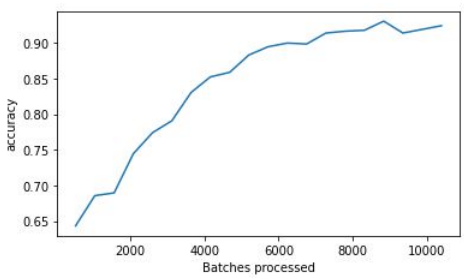
Épocas	Perdidas de entrenamiento	Perdidas de validación	Exactitud	Tasa de error
0	1.90	1.12	0.60	0.40
4	1.11	0.65	0.76	0.24
9	0.82	0.40	0.86	0.14
14	0.66	0.33	0.89	0.11
19	0.57	0.31	0.90	0.10

Tabla 3.5: Resultados de entrenamiento ResNet34
Fuente: autor

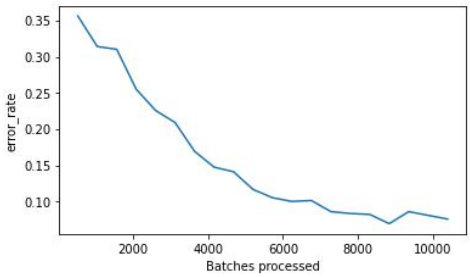
Los resultados reflejan que el modelo es entrenado de forma adecuada con una exactitud del 90 %, y una tasa de error del 10 %,

3.2.2 Entrenamiento con modelo ResNet50

Se puede visualizar el historial de entrenamiento de la exactitud y tasa de error en la Figura 3.11. Se puede observar que la red neuronal aprende de manera progresiva y que incrementar el número de capas en el modelo de red contribuye a un mejor ajuste durante el entrenamiento.



(a) Exactitud modelo ResNet50



(b) Tasa de error modelo ResNet50

Figura 3.11: Entrenamiento Resnet50

Se puede apreciar en la Figura 3.12 que los valores de pérdida disminuyen a medida que avanza el entrenamiento.

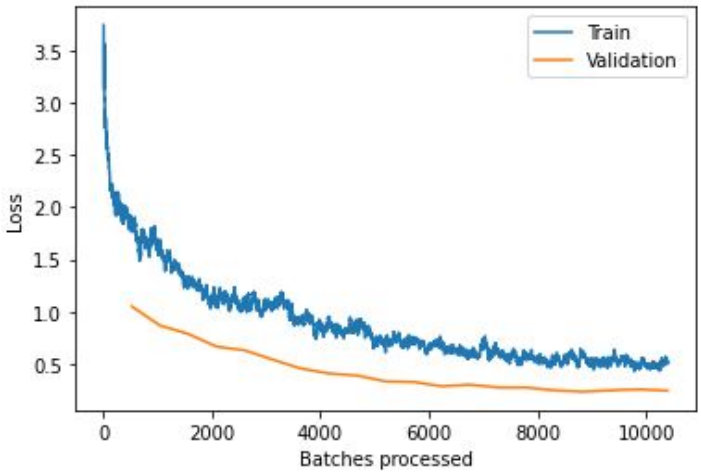


Figura 3.12: Pérdidas del modelo ResNet50

Épocas	Perdidas de entrenamiento	Perdidas de validación	Exactitud	Tasa de error
0	1.88	1.05	0.64	0.36
4	1.11	0.63	0.77	0.23
9	0.61	0.34	0.88	0.12
14	0.56	0.28	0.92	0.08
19	0.51	0.25	0.92	0.08

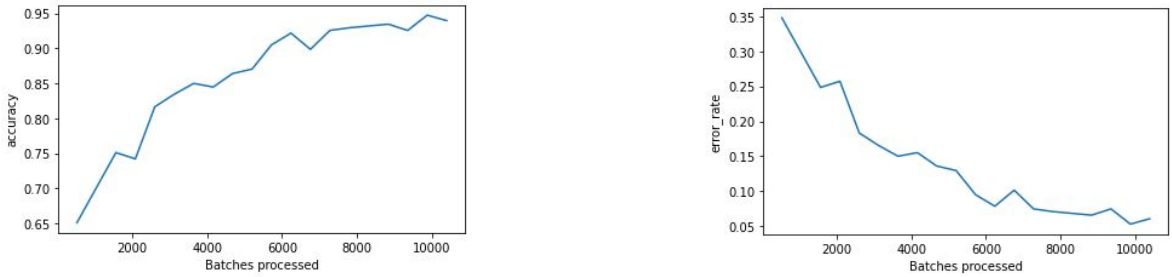
Tabla 3.6: Resultados de entrenamiento ResNet50
Fuente: autor

De la Tabla 3.6 se puede observar que se tiene una exactitud del 92 %, con una tasa

de error de 8 % ademas de disminuir las perdidas de entrenamiento y validación, eso quiere decir que mejora el entrenamiento del modelo con resnet34.

3.2.3 Entrenamiento con modelo ResNet101

Se muestran los resultados del entrenamiento del modelo ResNet101 mediante la Figura 3.13, donde se puede observar la gráfica de la exactitud versus los lotes procesados, así como la tasa de error durante las 20 épocas de entrenamiento.



(a) Exactitud modelo ResNet101 **(b)** Tasa de error modelo ResNet101

Figura 3.13: Entrenamiento Resnet101

Se puede apreciar en la Figura 3.13 que se obtiene una alta precisión, sin embargo, se presentan algunas fluctuaciones durante las épocas de entrenamiento.

Asimismo, en la Figura 3.14 se puede observar que los valores de pérdida disminuyen a medida que avanzan las épocas de entrenamiento.

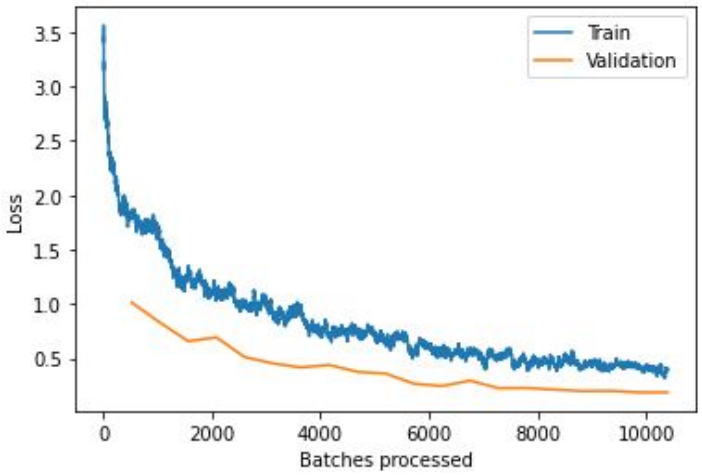


Figura 3.14: Resultados de entrenamiento ResNet50
Fuente: autor

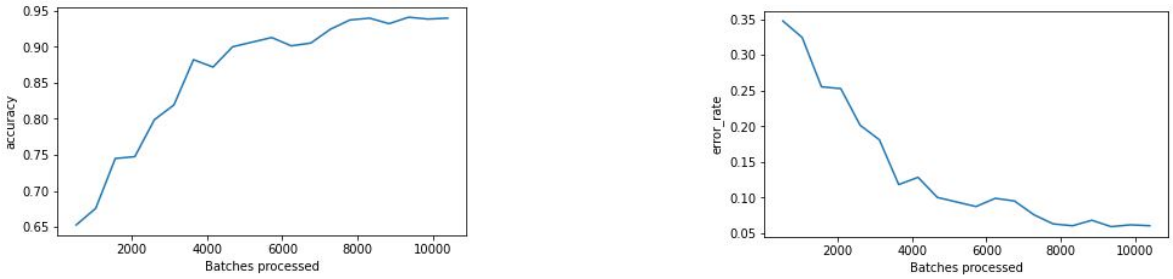
Épocas	Perdidas de entrenamiento	Perdidas de validación	Exactitud	Tasa de error
0	1.79	1.01	0.65	0.35
4	0.98	0.51	0.82	0.18
9	0.66	0.36	0.87	0.13
14	0.45	0.23	0.93	0.07
19	0.41	0.19	0.94	0.06

Tabla 3.7: Resultados de entrenamiento ResNet101
Fuente: autor

De la Tabla 3.7 se puede observar que se tiene una exactitud del 94 %, con una tasa de error de 6 % ademas de disminuir las perdidas de entrenamiento y validación, mejorando los resultados del modelo con resnet50.

3.2.4 Entrenamiento con modelo ResNet152

Se exponen los resultados del entrenamiento utilizando el modelo de red resnet152 en la Figura 3.15, donde se muestra la evolución de la exactitud y la tasa de error a lo largo de las 20 épocas de entrenamiento.



(a) Exactitud modelo ResNet152 **(b)** Tasa de error modelo ResNet152

Figura 3.15: Entrenamiento Resnet152

Se puede observar en la Figura 3.15 que se logra una buena exactitud, aunque se presentan oscilaciones similares al modelo resnet101 durante las épocas de entrenamiento.

Asimismo, en el gráfico de la Figura 3.16 se puede apreciar que los valores de pérdidas disminuyen a medida que avanzan las épocas de entrenamiento.

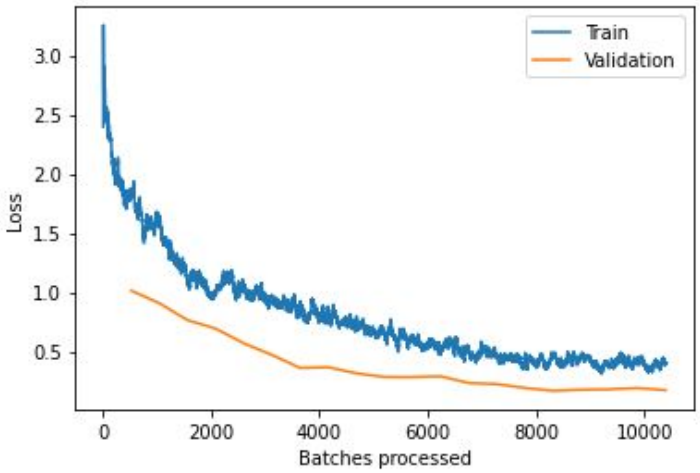


Figura 3.16: Pérdidas del modelo ResNet152

Épocas	Perdidas de entrenamiento	Perdidas de validación	Exactitud	Tasa de error
0	1.86	1.01	0.65	0.35
4	1.05	0.57	0.80	0.20
9	0.64	0.28	0.91	0.09
14	0.49	0.19	0.94	0.06
19	0.40	0.17	0.94	0.06

Tabla 3.8: Resultados de entrenamiento ResNet152
Fuente: autor

De la Tabla 3.8 se puede observar que se tiene una exactitud del 94 %, con una tasa de error de 6 %, se puede apreciar que este modelo tiene similitudes de resultados con el modelo resnet101.

3.3 ANÁLISIS DE RESULTADOS

Ya que se trata de un problema de clasificación, con el fin de obtener una evaluación y comprensión más precisa del modelo, se construye una matriz de confusión para cada modelo que se ha entrenado.

3.3.1 Análisis con modelo ResNet34

La Figura 3.17 presenta una matriz de confusión que muestra seis clases correspondientes a un total de 780 imágenes de prueba.

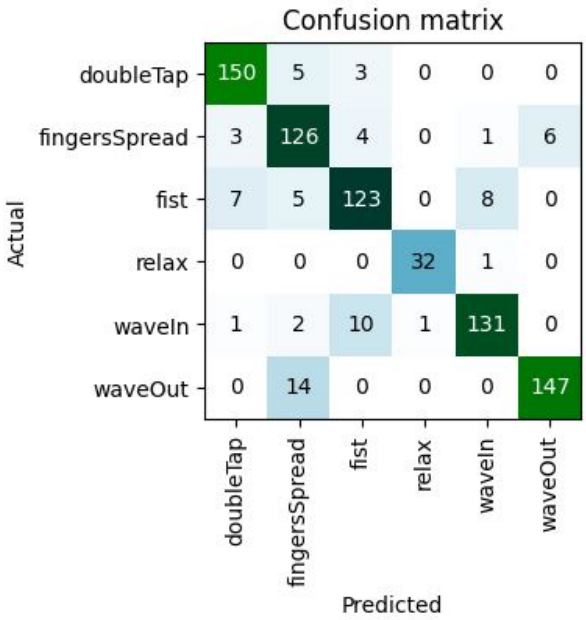


Figura 3.17: Matriz de confusión resnet34

Por consiguiente, se puede obtener una matriz de observación a partir de la matriz de confusión obtenida, en la que se muestran los valores de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) para cada clase como se observa en la Tabla 3.9. y poder calcular las métricas de cada clase que definirán el rendimiento del modelo.

Matriz de observacion				
Clases	**Medidas**			
	TP	**TN**	**FP**	**FN**
doubleTap	150	611	11	8
fingersSpread	126	614	26	14
fist	123	620	17	20
relax	32	746	1	1
waveIn	131	625	10	14
waveOut	147	613	6	14

Tabla 3.9: Matriz de observación resnet34
Fuente: autor

Después de obtener las medidas de TP, TN, FP y FN, se procede a calcular los indicadores correspondientes para cada clase de gesto. Los resultados de las métricas de clasificación se resumen en la Tabla 3.10, calculados con las ecuaciones pertinentes descritas en la sección 1.5.8.

Clases	Exactitud	Precision	Recall	F1 score
doubleTap	97.56	93.17	94.94	94.04
fingersSpread	94.87	82.89	90.00	86.30
fist	95.26	87.86	86.01	86.93
relax	99.74	96.97	96.97	96.97
waveIn	96.92	92.91	90.34	91.61
waveOut	97.44	96.08	91.30	93.63

Tabla 3.10: Métricas de Rendimiento resnet34
Fuente: autor

Como se observa en la Tabla 3.10 el valor de la exactitud de predicción para una red resnet34 entrenada con 20 épocas de entrenamiento es superior 94 % para cada clase, pero se puede observar que las precisiones, recall, F1 score mas bajas se tiene en los gestos fingesSpread y fist.

Profundizando más en las predicciones incorrectas, se puede obtener una muestra de imágenes que tuvieron la mayor pérdida del conjunto de datos. (Cuanto mayor

sea la pérdida, peor será el rendimiento del modelo). Ejecutando el comando plot_top_losses en google colab nos muestra en la Figura 3.18 una parte de las imágenes.

Cada imagen está etiquetada con cuatro cosas: predicción, actual (etiqueta objetivo), pérdida y probabilidad. La probabilidad aquí es el nivel de confianza, de cero a uno, que el modelo ha asignado a la predicción

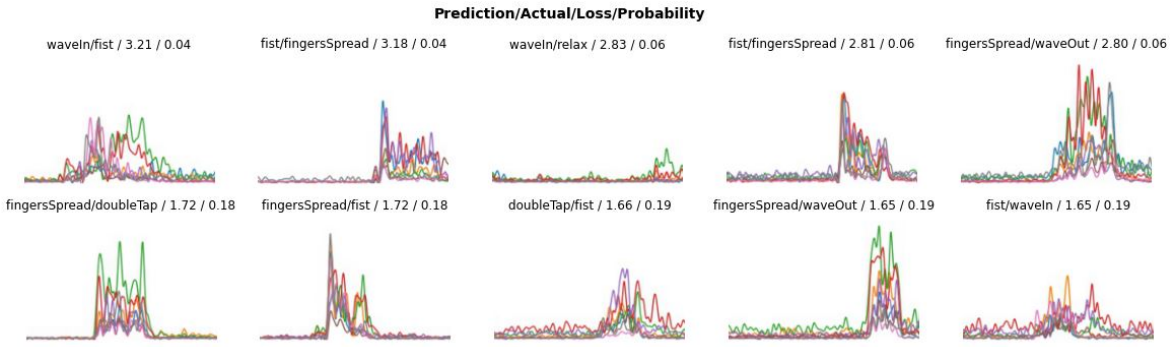


Figura 3.18: Top losses

La Tabla 3.11 permite identificar los gestos que presentan mayores errores de clasificación a partir de la matriz de confusión.

Gestos		**Cantidad**
Actual	**Predicción**	
waveOut	fingersSpread	14
waveIn	fist	10
fist	waveIn	8
fist	doubleTap	7

Tabla 3.11: Confusión de clasificación de gestos resnet34
Fuente: autor

3.3.2 Análisis con modelo ResNet50

La Figura 3.19 muestra la matriz de confusión resultante del uso del modelo resnet50, la cual contiene un total de 780 imágenes de prueba con sus respectivas seis clases.

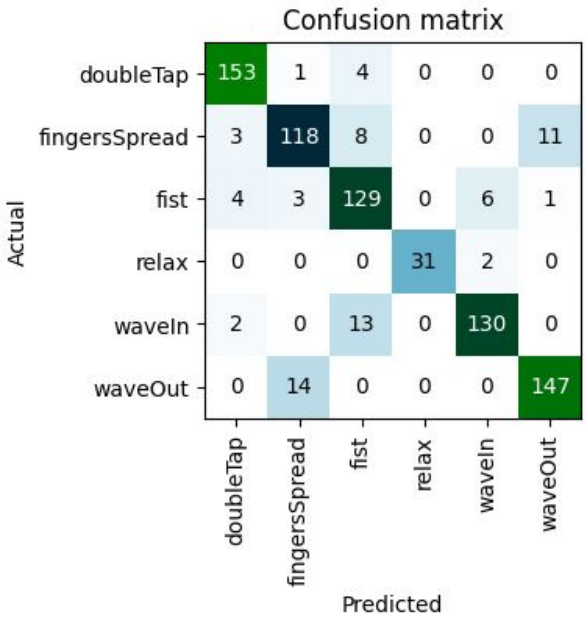


Figura 3.19: Matriz de confusión resnet50

A partir de la matriz de confusión obtenida con el modelo resnet50, se puede generar una matriz de observación que incluye los valores de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) para cada una de las seis clases, como se muestra en la Tabla 3.12.

Matriz de observación				
Clases	**Medidas**			
	TP	**TN**	**FP**	**FN**
doubleTap	153	613	9	5
fingersSpread	118	622	18	22
fist	129	612	25	14
relax	31	747	0	2
waveIn	130	627	8	15
waveOut	147	607	12	14

Tabla 3.12: Matriz de observación resnet50
Fuente: autor

Después de obtener las medidas de TP, TN, FP y FN, se calculan los indicadores correspondientes para cada gesto. La Tabla 3.13 resume los resultados de las métricas de clasificación, aplicando las ecuaciones correspondientes de la sección

1.5.8.

Clases	Exactitud	Precision	Recall	F1 score
doubleTap	98.21	94.44	96.84	95.63
fingersSpread	94.87	86.76	84.29	85.51
fist	95.00	83.77	90.21	86.87
relax	99.74	100.00	93.94	96.88
waveIn	97.05	94.20	89.66	91.87
waveOut	96.67	92.45	91.30	91.88

Tabla 3.13: Métricas de Rendimiento resnet50
Fuente: autor

Como se observa en la Tabla 3.13 el valor de la exactitud de predicción para una red resnet50 es superior 94 % para cada clase, las precisiones, recall, F1 score mas bajas se tiene en los gestos fingesSpread y fist con valores similares al modelo resnet34.

Se obtiene una muestra de imágenes que tuvieron la mayor pérdida del conjunto de datos. (Cuanto mayor sea la pérdida, peor será el rendimiento del modelo).

En la Figura 3.20 se muestra que cada imagen se ha etiquetado con cuatro elementos: predicción, etiqueta real (objetivo), pérdida y probabilidad. La probabilidad representa el nivel de confianza del modelo en la predicción, y varía de cero a uno.

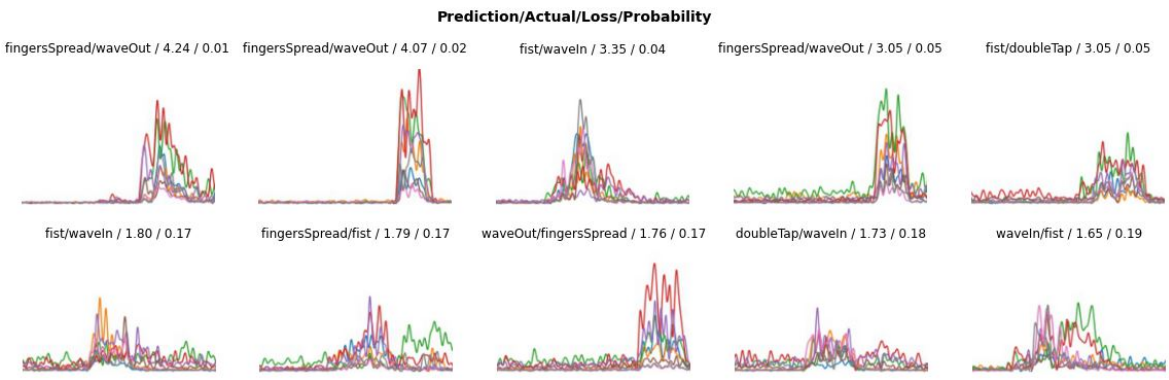


Figura 3.20: Top losses resnet50

La Tabla 3.14 permite identificar los gestos que presentan mayores errores de clasificación a partir de la matriz de confusión.

Gestos		Cantidad
Actual	**Predicción**	
waveOut	fingersSpread	14
waveIn	fist	13
fingersSpread	waveOut	11
fingersSpread	fist	8

Tabla 3.14: Confusión de clasificación de gestos resnet50
Fuente: autor

3.3.3 Análisis con modelo ResNet101

La Figura 3.21 muestra el resultado obtenido utilizando el modelo resnet101 en un total de 780 imágenes de prueba, cada una etiquetada con su respectiva clase entre un total de seis.

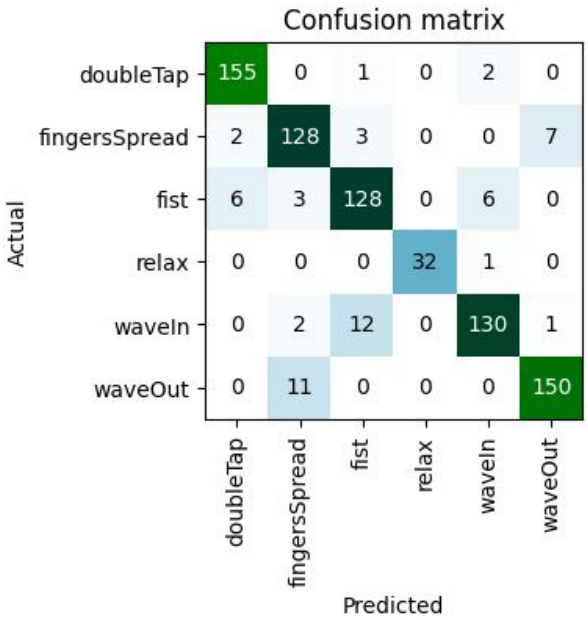


Figura 3.21: Matriz de confusión resnet101

A partir de la matriz de confusión obtenida con el modelo resnet101, se puede generar una matriz de observación que incluye los valores de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) para cada una de las seis clases, como se muestra en la Tabla 3.15

Matriz de observación				
Clases	**Medidas**			
	TP	**TN**	**FP**	**FN**
doubleTap	155	614	8	3
fingersSpread	128	624	16	12
fist	128	621	16	15
relax	32	747	0	1
waveIn	130	626	9	15
waveOut	150	611	8	11

Tabla 3.15: Matriz de observación resnet101
Fuente: autor

Después de obtener las medidas de TP, TN, FP y FN, se calculan los indicadores correspondientes para cada gesto. La Tabla 3.16 resume los resultados de las métricas de clasificación, aplicando las ecuaciones correspondientes de la sección 1.5.8.

Clases	Exactitud	Precision	Recall	F1 score
doubleTap	98.59	95.09	98.10	96.57
fingersSpread	96.41	88.89	91.43	90.14
fist	96.03	88.89	89.51	89.20
relax	99.87	100.00	96.97	98.46
waveIn	96.92	93.53	89.66	91.55
waveOut	97.56	94.94	93.17	94.04

Tabla 3.16: Métricas de Rendimiento resnet101
Fuente: autor

Como se observa en la Tabla 3.16 el valor de la exactitud de predicción para una red resnet101 es superior 96 % para cada clase, los valores de precisión, recall, F1 score son mejores que los modelos anteriores.

Se obtiene una muestra de imágenes que tuvieron la mayor pérdida del conjunto de datos. (Cuanto mayor sea la pérdida, peor será el rendimiento del modelo).

Cada imagen está etiquetada con cuatro cosas: predicción, actual (etiqueta

objetivo), pérdida y probabilidad. La probabilidad aquí es el nivel de confianza, de cero a uno, que el modelo ha asignado a la predicción, como se puede apreciar en la Figura 3.22.

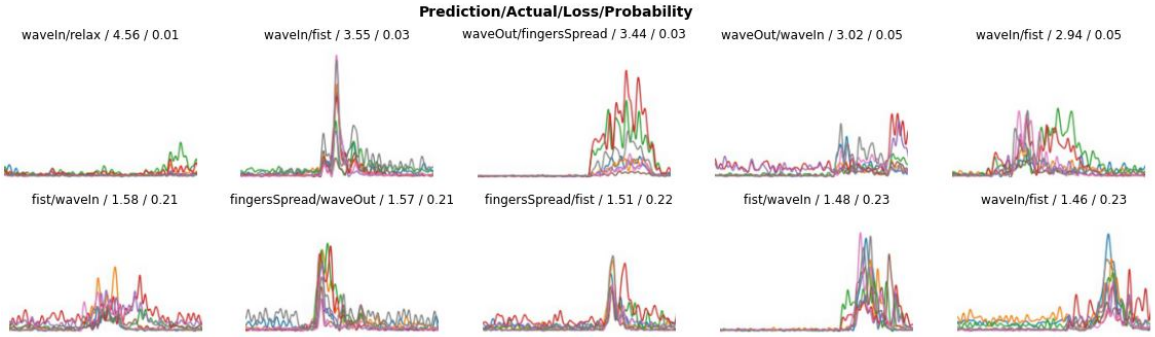


Figura 3.22: Top losses resnet101

La Tabla 3.17 permite identificar los gestos que presentan mayores errores de clasificación a partir de la matriz de confusión.

Gestos		**Cantidad**
Actual	**Predicción**	
waveIn	fist	12
waveOut	fingersSpread	11
fingersSpread	waveOut	7
fist	dobleTap	6

Tabla 3.17: Confusión de clasificación de gestos resnet101
Fuente: autor

3.3.4 Análisis con modelo ResNet152

La Figura 3.23 muestra la matriz de confusión resultante del uso del modelo resnet152, la cual contiene un total de 780 imágenes de prueba con sus respectivas seis clases.

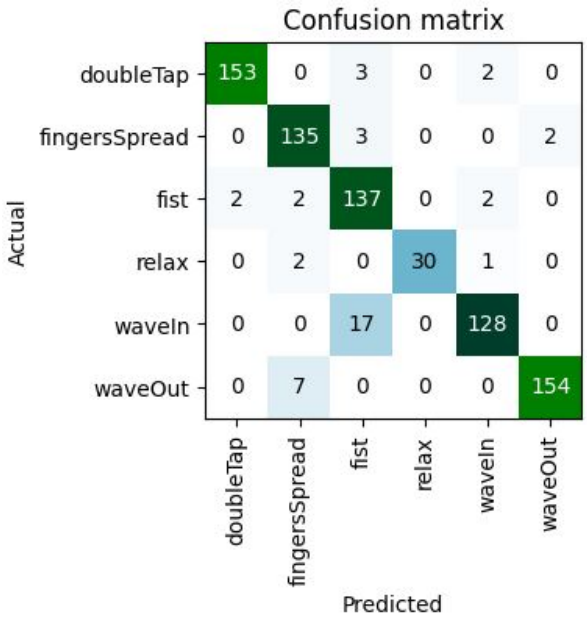


Figura 3.23: Matriz de confusión resnet152

A partir de la matriz de confusión obtenida con el modelo resnet152, se puede generar una matriz de observación que incluye los valores de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) para cada una de las seis clases, como se muestra en la Tabla 3.18

Matriz de observación				
Clases	**Medidas**			
	TP	**TN**	**FP**	**FN**
doubleTap	153	620	2	5
fingersSpread	135	629	11	5
fist	137	614	23	6
relax	30	747	0	3
waveIn	128	630	5	17
waveOut	154	617	2	7

Tabla 3.18: Matriz de observación resnet152
Fuente: autor

Después de obtener las medidas de TP, TN, FP y FN, se calculan los indicadores correspondientes para cada gesto. La Tabla 3.19 resume los resultados de las métricas de clasificación, aplicando las ecuaciones correspondientes de la sección

1.5.8.

Clases	Exactitud	Precision	Recall	F1 score
doubleTap	99.10	98.71	96.84	97.76
fingersSpread	97.95	92.47	96.43	94.41
fist	96.28	85.63	95.80	90.43
relax	99.62	100.00	90.91	95.24
waveIn	97.18	96.24	88.28	92.09
waveOut	98.85	98.72	95.65	97.16

Tabla 3.19: Métricas de Rendimiento resnet152
Fuente: autor

Como se observa en la Tabla 3.19 el valor de la exactitud de predicción para una red resnet152 es superior 96 % para cada clase, la precisión del gesto fist y recall del gesto waveIn son los valores mas bajos.

Se obtiene una muestra de imágenes que tuvieron la mayor pérdida del conjunto de datos. (Cuanto mayor sea la pérdida, peor será el rendimiento del modelo).

Cada imagen está etiquetada con cuatro cosas: predicción, actual (etiqueta objetivo), pérdida y probabilidad. La probabilidad aquí es el nivel de confianza, de cero a uno, que el modelo ha asignado a la predicción, como se puede apreciar en la Figura 3.24.

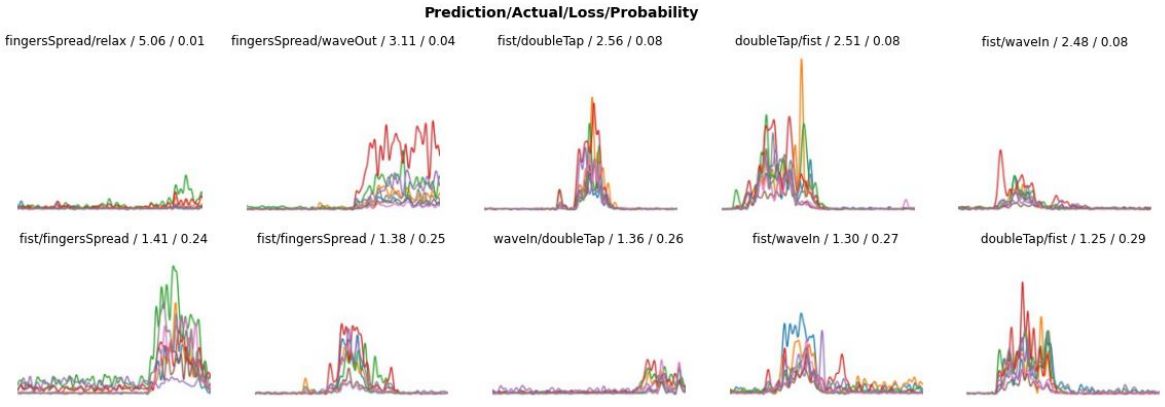


Figura 3.24: Top losses resnet152

La Tabla 3.20 permite identificar los gestos que presentan mayores errores de clasificación a partir de la matriz de confusión.

Gestos		Cantidad
Actual	**Predicción**	
waveIn	fist	17
waveOut	fingersSpread	7
dobleTap	fist	3
fingersSpread	fist	3

Tabla 3.20: Confusión de clasificación de gestos resnet152
Fuente: autor

3.4 ANÁLISIS DE RESULTADOS CON DATOS SIN PREPROCESAR

Ya que se trata de un problema de clasificación, con el fin de obtener una evaluación y comprensión más precisa del modelo, se construye una matriz de confusión para cada modelo que se ha entrenado con datos sin preprocesar.

3.4.1 Análisis con modelo ResNet34

La Figura 3.25 presenta una matriz de confusión que muestra seis clases correspondientes a un total de 780 imágenes de prueba.

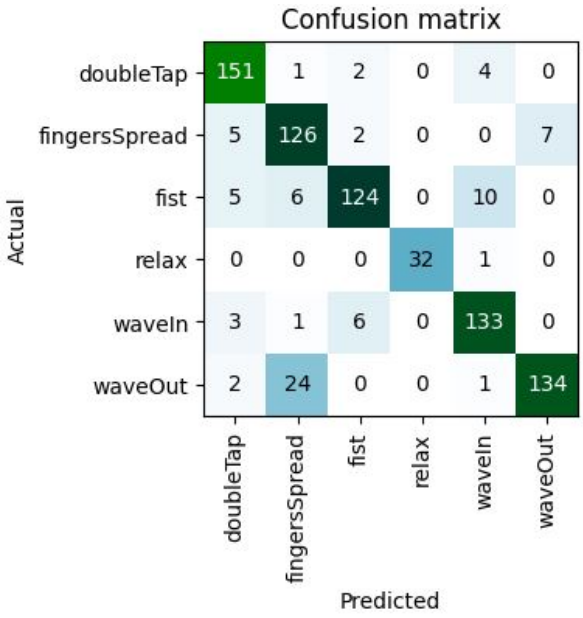


Figura 3.25: Matriz de confusión resnet34

A partir de la matriz de confusión obtenida con el modelo resnet34, se puede generar una matriz de observación que incluye los valores de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) para cada una de las seis clases, como se muestra en la Tabla 3.21. y poder calcular las métricas de cada clase que definirán el rendimiento del modelo.

Matriz de observacion				
Clases	**Medidas**			
	TP	**TN**	**FP**	**FN**
doubleTap	151	607	15	7
fingersSpread	126	608	32	14
fist	124	625	10	21
relax	32	747	0	1
waveIn	133	621	16	10
waveOut	134	612	7	27

Tabla 3.21: Matriz de observación
Fuente: autor

Después de obtener las medidas de TP, TN, FP y FN, se calculan los indicadores correspondientes para cada gesto. La Tabla 3.22 resume los resultados de las métricas de clasificación, aplicando las ecuaciones correspondientes de la sección 1.5.8.

Clases	**Exactitud**	**Precision**	**Recall**	**F1 score**
doubleTap	97.18	90.96	95.57	93.21
fingersSpread	94.10	79.75	90.00	84.56
fist	96.03	92.54	85.52	88.89
relax	99.87	100.00	96.97	98.46
waveIn	96.67	89.26	93.01	91.10
waveOut	95.64	95.04	83.23	88.74

Tabla 3.22: Métricas de Rendimiento resnet34
Fuente: autor

Como se observa en la Tabla 3.22 el valor de la exactitud de predicción para una red resnet34 entrenada con 20 épocas de entrenamiento es superior 94 % para cada

clase, pero se puede observar que las precisiones mas bajas se tiene en los gestos fingesSpread y waveIn.

Profundizando más en las predicciones incorrectas, se puede obtener una muestra de imágenes que tuvieron la mayor pérdida del conjunto de datos. (Cuanto mayor sea la pérdida, peor será el rendimiento del modelo). Ejecutando el comando plot_top_losses en google colab nos muestra en la Figura 3.26 una parte de las imágenes.

Cada imagen está etiquetada con cuatro cosas: predicción, actual (etiqueta objetivo), pérdida y probabilidad. La probabilidad aquí es el nivel de confianza, de cero a uno, que el modelo ha asignado a la predicción

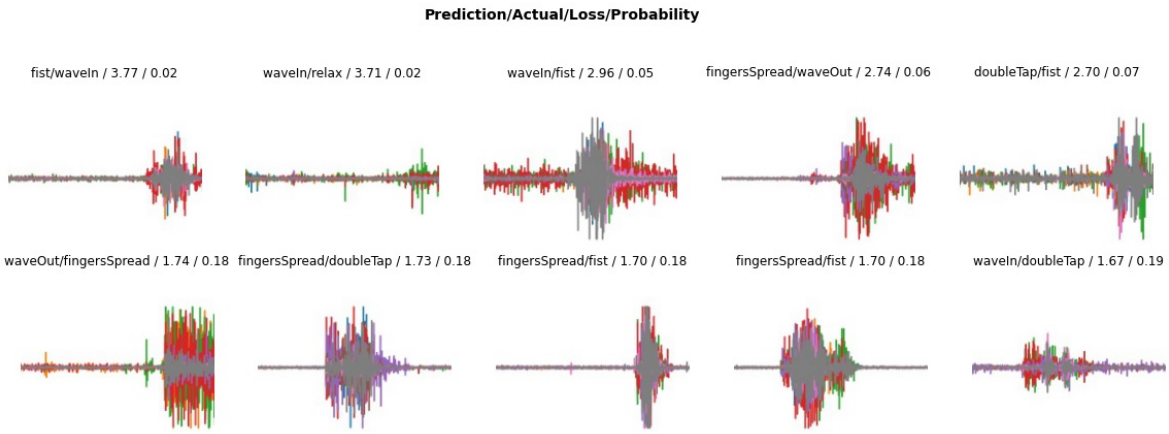


Figura 3.26: Top losses

La Tabla 3.23 permite identificar los gestos que presentan mayores errores de clasificación a partir de la matriz de confusión.

Gestos		**Cantidad**
Actual	**Predicción**	
waveOut	fingersSpread	24
fist	waveIn	10
fingersSpread	waveOut	7
fist	fingersSpread	6

Tabla 3.23: Confusión de clasificación de gestos resnet34
Fuente: autor

3.4.2 Análisis con modelo ResNet50

La Figura 3.27 presenta una matriz de confusión que muestra seis clases correspondientes a un total de 780 imágenes de prueba.

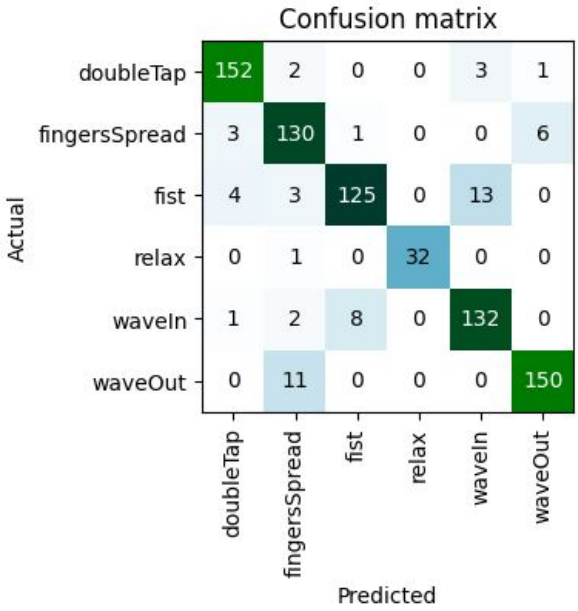


Figura 3.27: Matriz de confusión resnet50

A partir de la matriz de confusión obtenida con el modelo resnet50, se puede generar una matriz de observación que incluye los valores de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) para cada una de las seis clases, como se muestra en la Tabla 3.24.

Matriz de observación				
Clases	**Medidas**			
	TP	**TN**	**FP**	**FN**
doubleTap	152	614	8	6
fingersSpread	130	621	19	10
fist	125	626	9	20
relax	32	747	0	1
waveIn	132	621	16	11
waveOut	150	612	7	11

Tabla 3.24: Matriz de observación resnet50
Fuente: autor

Después de obtener las medidas de TP, TN, FP y FN, se calculan los indicadores correspondientes para cada gesto. La Tabla 3.25 resume los resultados de las métricas de clasificación, aplicando las ecuaciones correspondientes de la sección 1.5.8.

Clases	Exactitud	Precision	Recall	F1 score
doubleTap	98.21	95.00	96.20	95.60
fingersSpread	96.28	87.25	92.86	89.97
fist	96.28	93.28	86.21	89.61
relax	99.87	100.00	96.97	98.46
waveIn	96.54	89.19	92.31	90.72
waveOut	97.69	95.54	93.17	94.34

Tabla 3.25: Métricas de Rendimiento resnet50
Fuente: autor

Como se observa en la Tabla 3.25 el valor de la exactitud de predicción para una red resnet50 es superior 96 % para cada clase, la precisión del gesto fingersSpread y recall del gesto fist son los valores mas bajos.

Se obtiene una muestra de imágenes que tuvieron la mayor pérdida del conjunto de datos. (Cuanto mayor sea la pérdida, peor será el rendimiento del modelo).

Cada imagen está etiquetada con cuatro cosas: predicción, actual (etiqueta

objetivo), pérdida y probabilidad. La probabilidad aquí es el nivel de confianza, de cero a uno, que el modelo ha asignado a la predicción, como se puede apreciar en la Figura 3.28.

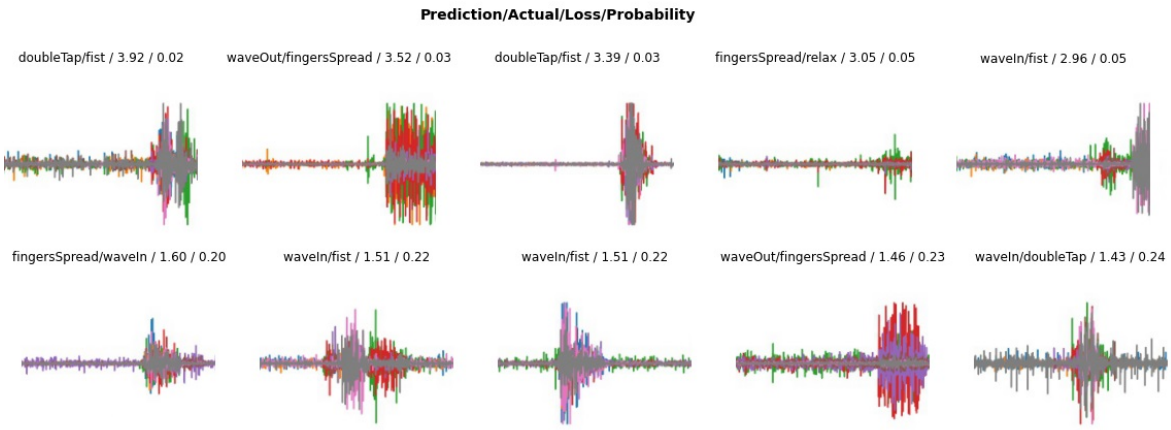


Figura 3.28: Top losses resnet50

La Tabla 3.26 permite identificar los gestos que presentan mayores errores de clasificación a partir de la matriz de confusión.

Gestos		Cantidad
Actual	**Predicción**	
fist	waveIn	13
waveOut	fingersSpread	11
waveIn	fist	8
fingersSpread	waveOut	6

Tabla 3.26: Confusión de clasificación de gestos resnet50
Fuente: autor

3.4.3 Análisis con modelo ResNet101

La Figura 3.29 presenta una matriz de confusión que muestra seis clases correspondientes a un total de 780 imágenes de prueba.

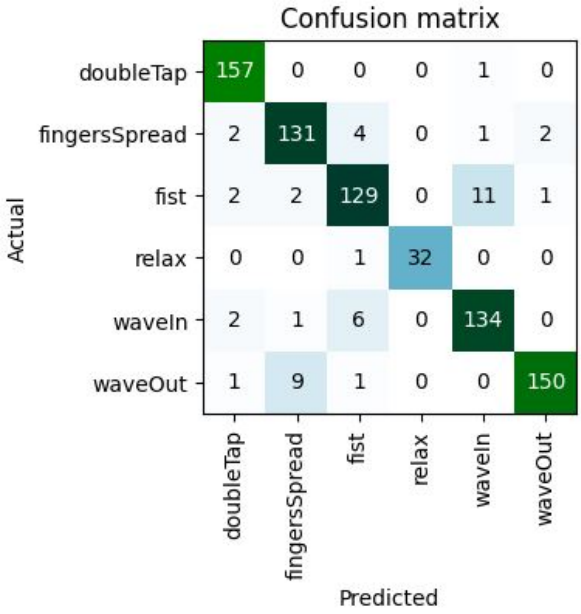


Figura 3.29: Matriz de confusión resnet101

A partir de la matriz de confusión obtenida con el modelo resnet101, se puede generar una matriz de observación que incluye los valores de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) para cada una de las seis clases, como se muestra en la Tabla Tabla 3.27.

Matriz de observacion				
Clases	**Medidas**			
	TP	**TN**	**FP**	**FN**
doubleTap	157	615	7	1
fingersSpread	131	628	12	9
fist	129	623	12	16
relax	32	747	0	1
waveIn	134	624	13	9
waveOut	150	616	3	11

Tabla 3.27: Matriz de observación resnet101
Fuente: autor

Después de obtener las medidas de TP, TN, FP y FN, se calculan los indicadores correspondientes para cada gesto. La Tabla 3.28 resume los resultados de las métricas de clasificación, aplicando las ecuaciones correspondientes de la sección

1.5.8.

Clases	Exactitud	Precision	Recall	F1 score
doubleTap	98.97	95.73	99.37	97.52
fingersSpread	97.31	91.61	93.57	92.58
fist	96.41	91.49	88.97	90.21
relax	99.87	100.00	96.97	98.46
waveIn	97.18	91.16	93.71	92.41
waveOut	98.21	98.04	93.17	95.54

Tabla 3.28: Métricas de Rendimiento resnet101
Fuente: autor

Como se observa en la Tabla 3.28 el valor de la exactitud de predicción para una red resnet101 es superior 96 % para cada clase, el valor de recall del gesto fist es el mas bajo.

Se obtiene una muestra de imágenes que tuvieron la mayor pérdida del conjunto de datos. (Cuanto mayor sea la pérdida, peor será el rendimiento del modelo).

Cada imagen está etiquetada con cuatro cosas: predicción, actual (etiqueta objetivo), pérdida y probabilidad. La probabilidad aquí es el nivel de confianza, de cero a uno, que el modelo ha asignado a la predicción, como se puede apreciar en la Figura 3.30.

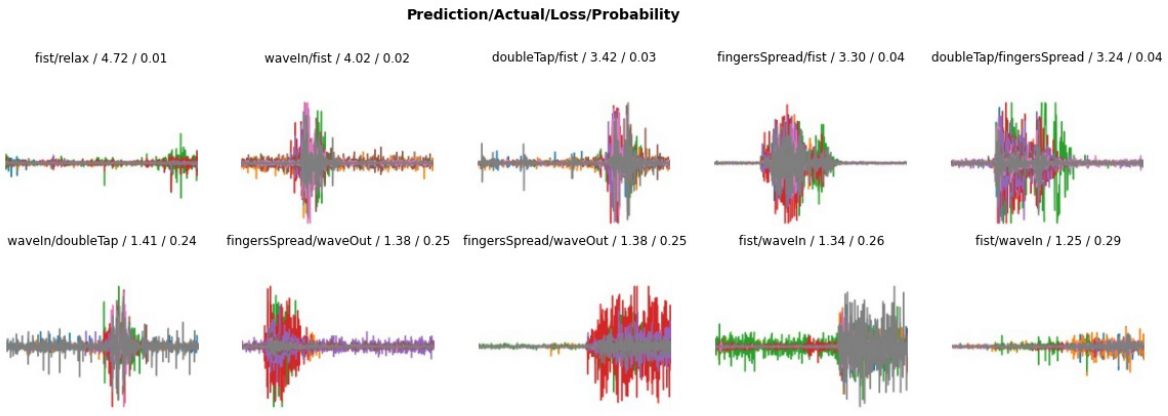


Figura 3.30: Top losses resnet101

La Tabla 3.29 permite identificar los gestos que presentan mayores errores de

clasificación a partir de la matriz de confusión.

Gestos		Cantidad
Actual	**Predicción**	
fist	waveIn	11
waveOut	fingersSpread	9
waveIn	fist	6
fingersSpread	fist	4

Tabla 3.29: Confusión de clasificación de gestos resnet101
Fuente: autor

3.4.4 Análisis con modelo ResNet152

La Figura 3.31 presenta una matriz de confusión que muestra seis clases correspondientes a un total de 780 imágenes de prueba.

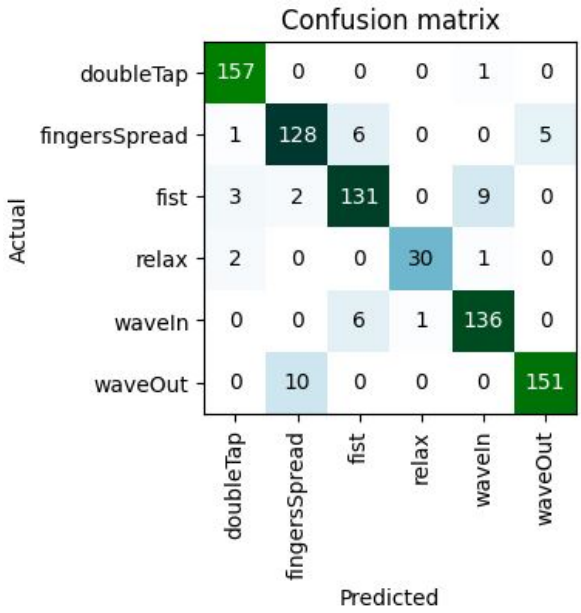


Figura 3.31: Matriz de confusión resnet152

A partir de la matriz de confusión obtenida con el modelo resnet152, se puede generar una matriz de observación que incluye los valores de verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN) para cada una de las seis clases, como se muestra en la Tabla 3.30.

Matriz de observacion				
Clases	**Medidas**			
	TP	**TN**	**FP**	**FN**
doubleTap	157	616	6	1
fingersSpread	128	628	12	12
fist	131	623	12	14
relax	30	746	1	3
waveIn	136	626	11	7
waveOut	151	614	5	10

Tabla 3.30: Matriz de observación resnet152
Fuente: autor

Después de obtener las medidas de TP, TN, FP y FN, se calculan los indicadores correspondientes para cada gesto. La Tabla 3.31 resume los resultados de las métricas de clasificación, aplicando las ecuaciones correspondientes de la sección 1.5.8.

Clases	Exactitud	Precision	Recall	F1 score
doubleTap	99.10	96.32	99.37	97.82
fingersSpread	96.92	91.43	91.43	91.43
fist	96.67	91.61	90.34	90.97
relax	99.49	96.77	90.91	93.75
waveIn	97.69	92.52	95.10	93.79
waveOut	98.08	96.79	93.79	95.27

Tabla 3.31: Métricas de Rendimiento resnet152
Fuente: autor

Como se observa en la Tabla 3.31 el valor de la exactitud de predicción para una red resnet152 es superior 96 % para cada clase, la precisión, recall, F1 score superan el 90 %.

Se obtiene una muestra de imágenes que tuvieron la mayor pérdida del conjunto de datos. (Cuanto mayor sea la pérdida, peor será el rendimiento del modelo).

Cada imagen está etiquetada con cuatro cosas: predicción, actual (etiqueta

objetivo), pérdida y probabilidad. La probabilidad aquí es el nivel de confianza, de cero a uno, que el modelo ha asignado a la predicción, como se puede apreciar en la Figura 3.32.

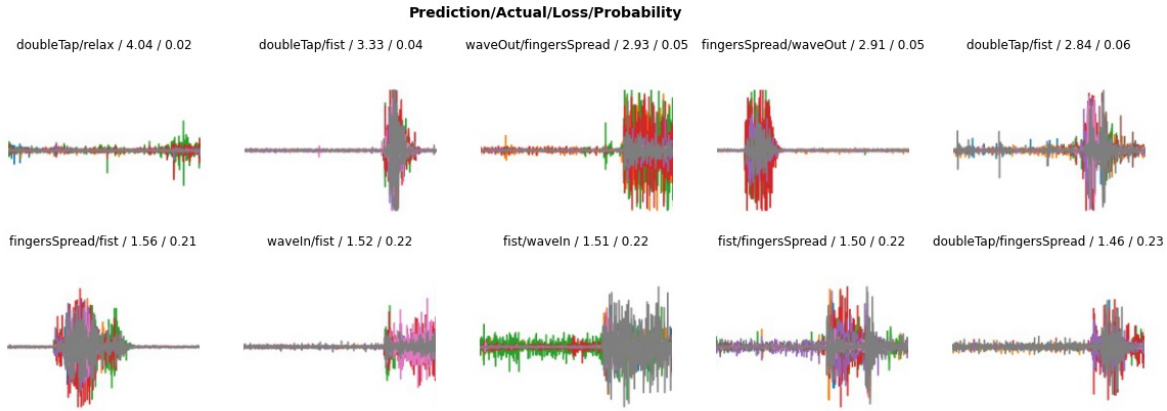


Figura 3.32: Top losses resnet152

La Tabla 3.32 permite identificar los gestos que presentan mayores errores de clasificación a partir de la matriz de confusión.

Gestos		**Cantidad**
Actual	**Predicción**	
waveOut	fingersSpread	10
fist	waveIn	9
fingersSpread	fist	6
waveIn	fist	6

Tabla 3.32: Confusión de clasificación de gestos resnet152
Fuente: autor

4 CONCLUSIONES

- La implementación de un algoritmo de inteligencia artificial en Google Colab es un proceso complejo que requiere un gran conocimiento en el área de aprendizaje automático y procesamiento de datos. No obstante, utilizando las herramientas y recursos disponibles en Google Colab, es posible desarrollar e implementar un algoritmo de IA de manera eficiente y efectiva. Además, la plataforma ofrece recursos para el procesamiento de datos y la ejecución de modelos de aprendizaje profundo o deep learning, lo que ayuda en el desarrollo y la implementación de algoritmos de IA. Es importante también recordar que la implementación de IA no es solo un proceso de programación, sino también de análisis y comprensión de los datos, metodología y modelos elegidos.

- Una de los beneficios mas importantes al implementar un modelo de IA con Google Colab es que posibilita el uso de una GPU que resulta mas potente que en una CPU, anteriormente tomaban días en realizar un entrenamiento de red, pero ahora se resuelven en horas o incluso minutos dependiendo la complejidad.

- Organizar una base de datos en un data frame es un paso crucial para el análisis de datos, ya que facilita su manipulación de forma estructurada y accesible. La aplicación de técnicas de Big Data permite el manejo eficiente y efectivo de grandes volúmenes de datos. Asimismo, el uso de herramientas como pandas, posibilita el procesamiento y análisis de datos y la generación de data frames para su posterior estudio. Es importante también asegurar la calidad de los datos, tanto en la recolección como en el procesamiento y limpieza, para obtener conclusiones confiables y robustas.

- En este trabajo se ha propuesto un modelo de CNN basado en la arquitectura Resnet comparando con varias capas para clasificar los 5 gestos de la mano usando señales emg utilizando un dataset publico, al examinar los resultados del entrenamiento, podemos decir que todos los resultados obtenidos han logrado una precisión muy efectiva en la clasificación de los gestos de la mano.

- La utilización de CNNs permite adquirir representaciones complejas de los datos y mejorar la exactitud en la categorización o clasificación de los movimientos de la mano. Además, las CNNs tienen una excelente capacidad para aprender patrones espaciales en los datos, lo que las hace ideales para clasificar señales EMG.

- El modelo basado en CNN propuesto proporcionó una mayor precisión incluso sin un preprocesamiento complejo y extracción de características en comparación con los métodos tradicionales utilizados con frecuencia. Los resultados obtenidos indicaron que el método de aprendizaje profundo propuesto podría ofrecer una mejor predicción del gesto realizado.

- Es fundamental evaluar el rendimiento del modelo mediante un conjunto de métricas adecuadas para poder determinar su capacidad de generalización y su efectividad en la tarea específica para la cual fue diseñado.

5 REFERENCIAS BIBLIOGRÁFICAS

Adaloglou, N. . (2021, 21 de enero). *Best deep CNN architectures and their principles: from AlexNet to EfficientNet*. Consultado el 5 de julio de 2022, desde https://theaisummer.com/cnn-architectures/#resnet-deep-residual-learning-for-image-recognition-2015

Asghari Oskoei, M., & Hu, H. (2007). Myoelectric control systemsA survey. *Biomedical Signal Processing and Control*, *2*(4), 275-294. https://doi.org/https://doi.org/10.1016/j.bspc.2007.07.009

Bisong, E. (2019a). Convolutional Neural Networks (CNN). En *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners* (pp. 423-441). Apress. https://doi.org/10.1007/978-1-4842-4470-8_35

Bisong, E. (2019b). Neural Network Foundations. En *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners* (pp. 331-332). Apress. https://doi.org/10.1007/978-1-4842-4470-8_28

Bisong, E. (2019c). What Is Cloud Computing? En *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners* (pp. 3-6). Apress. https://doi.org/10.1007/978-1-4842-4470-8_1

Borja-Robalino, R., Monleon-Getino, A., & Benedé, J. (2020). Estandarización de Métricas de Rendimiento para Clasificadores Machine y Deep Learning.

Carneiro, T., Nóbrega, R. V. M. D., Nepomuceno, T., Bian, G., de Albuquerque, V. H. C., & Filho, P. (2018). Performance Analysis of Google Colaboratory

as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, *6*, 61677-61685.

Castello, E. (2017). A wearable general-purpose solution for Human-Swarm Interaction.

Chen, L., Fu, J., Wu, Y., Li, H., & Zheng, B. (2020). Hand Gesture Recognition Using Compact CNN via Surface Electromyography Signals. *Sensors*, *20*(3). https://doi.org/10.3390/s20030672

Colab, G. (s.f.). Colaboratory Preguntas frecuentes. https://research.google.com/colaboratory/intl/es/faq.html

Côté-Allard, U., Fall, C. L., Campeau-Lecours, A., Gosselin, C., Laviolette, F., & Gosselin, B. (2017). Transfer learning for sEMG hand gestures recognition using convolutional neural networks. *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 1663-1668. https://doi.org/10.1109/SMC.2017.8122854

da Silva, I. N., Hernane Spatti, D., Andrade Flauzino, R., Liboni, L. H. B., & dos Reis Alves, S. F. (2017). Artificial Neural Network Architectures and Training Processes. En *Artificial Neural Networks : A Practical Course* (pp. 21-28). Springer International Publishing. https://doi.org/10.1007/978-3-319-43162-8_2

Flores, O. . J. . A. . (2021). Repositorio Digital - EPN: Implementación de un algoritmo para el reconocimiento de gestos de la mano en tiempo real, usando señales electromiográficas. https://bibdigital.epn.edu.ec/handle/15000/21698

Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for Multi-Class Classification: an Overview. https://doi.org/10.48550/ARXIV.2008.05756

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. https://doi.org/10.48550/ARXIV.1512.03385

Howard, J., & Gugger, S. (2020). *Deep Learning for Coders with Fastai and Pytorch: AI Applications Without a PhD*. O'Reilly Media, Incorporated. https://books.google.no/books?id=xd6LxgEACAAJ

Hudgins, B., Parker, P., & Scott, R. (1993). A new strategy for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering*, *40*(1), 82-94. https://doi.org/10.1109/10.204774

Iman, M., Arabnia, H., & Branchinst, R. (2020). Pathways to Artificial General Intelligence: A Brief Overview of Developments and Ethical Issues via Artificial Intelligence, Machine Learning, Deep Learning, and Data Science. https://doi.org/10.1007/978-3-030-70296-0

Keenan, K. G., & Enoka, R. M. (2012). Electromyography. En F. C. Mooren (Ed.), *Encyclopedia of Exercise Medicine in Health and Disease* (pp. 276-279). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-29807-6_298

Konrad, P. (2006). The ABC of EMG A Practical Introduction to Kinesiological Electromyography. https://www.noraxon.com/wp-content/uploads/2014/12/ABC-EMG-ISBN.pdf

Laboratorio de Investigación en Inteligencia y Visión Artificial. (2020). EMG-EPN-120 - Laboratorio de Investigación en Inteligencia y Visión Artificial. https://laboratorio-ia.epn.edu.ec/en/resources/dataset/2020_emg_dataset_120

Muñoz, J. (2019). Herramienta software para reconocimiento de objetos como ayuda a los procesos de mercadeo institucional. https://red.uao.edu.co/bitstream/handle/10614/11796/T08860.pdf?sequence=5&isAllowed=y

Nelli, F. (2018). *Python Data Analytics* (Vol. 2). Apress.

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. https://doi.org/10.48550/ARXIV.1511.08458

Ozdemir, M. A., Kisa, D. H., Guren, O., Onan, A., & Akan, A. (2020). EMG based Hand Gesture Recognition using Deep Learning. *2020 Medical Technologies Congress (TIPTEKNO)*, 1-4. https://doi.org/10.1109/TIPTEKNO50054.2020.9299264

Pajankar, A., & Joshi, A. (2022). Convolutional Neural Networks. En *Hands-on Machine Learning with Python: Implement Neural Network Solutions with Scikit-learn and PyTorch* (pp. 261-284). Apress. https://doi.org/10.1007/978-1-4842-7921-2_14

Phinyomark, A., Phukpattaranont, P., & Limsakul, C. (2012). Feature reduction and selection for EMG signal classification. *Expert Systems with Applications*, *39*(8), 7420-7431. https://doi.org/https://doi.org/10.1016/j.eswa.2012.01.102

Piña, J. (2017). Sistema inteligente de clasificación para apertura y cierre de la mano utilizando señales mioeléctricas de los músculos del antebrazo. https://repository.usta.edu.co/handle/11634/10958

Python.org. (2022). pickle Python object serialization Python 3.10.7 documentation. Consultado el 25 de agosto de 2022, desde https://docs.python.org/3/library/pickle.html

Silaparasetty, N. (2020a). Introduction to Deep Learning. En *Machine Learning Concepts with Python and the Jupyter Notebook Environment: Using Tensorflow 2.0* (pp. 41-56). Apress. https://doi.org/10.1007/978-1-4842-5967-2_3

Silaparasetty, N. (2020b). An Overview of Machine Learning. En *Machine Learning Concepts with Python and the Jupyter Notebook Environment: Using Tensorflow 2.0* (pp. 21-39). Apress. https://doi.org/10.1007/978-1-4842-5967-2_2

Sinthong, P., & Carey, M. J. (2019). AFrame: Extending DataFrames for Large-Scale Modern Data Analysis. *2019 IEEE International Conference on Big Data (Big Data)*, 359-371. https://doi.org/10.1109/BigData47090.2019.9006303

Smola, A., & Vishwanathan, S. (2008). Introduction to Machine Learning. http://alex.smola.org/drafts/thebook.pdf

Tapia, S. . R. . (2022). Repositorio de la Universidad de Fuerzas Armadas ESPE: Desarrollo de un clasificador de video para la detección automática de eventos de asalto a peatones basado en algoritmos de aprendizaje profundo. https://repositorio.espe.edu.ec/handle/21000/31604

Teoh, T. T., & Rong, Z. (2022). Convolutional Neural Networks. En *Artificial Intelligence with Python* (pp. 261-275). Springer Singapore. https://doi.org/10.1007/978-981-16-8615-3_16

Ting, K. M. (2010). Confusion Matrix. En C. Sammut & G. I. Webb (Eds.), *Encyclopedia of Machine Learning* (pp. 209-209). Springer US. https://doi.org/10.1007/978-0-387-30164-8_157

Universidad de las Fuerzas Armadas ESPE. (2020). Reconocimiento de Gestos de la Mano Usando Señales Electromiográficas (EMG) e Inteligencia Artificial. Consultado el 20 de abril de 2022, desde https://deee-el.espe.edu.ec/proyectos-de-investigacion-2019/proyecto-de-investigacion-1-2020/

Xu, Y., & Dai, Y. (2017). Review of hand gesture recognition study and application. *Contemporary engineering sciences*, *10*, 375-384.

Yalçn, O. G. (2021). Introduction to Machine Learning. En *Applied Neural Networks with TensorFlow 2: API Oriented Deep Learning with Python* (pp. 33-55). Apress. https://doi.org/10.1007/978-1-4842-6513-0_2

6 ANEXOS

6.1 ANEXO I

6.1.1 Creación y configuración Google Colab

Los siguientes pasos proporcionan un recorrido para iniciar un cuaderno de jupiter notebook en Google Colab:

1. Inicialmente se debe ingresar con un navegador web preferiblemente Google Chrome a https://colab.research.google.com/ e iniciar sesión con una cuenta de Google existente para acceder a la página de inicio de Colab, como indica la Figura 6.1

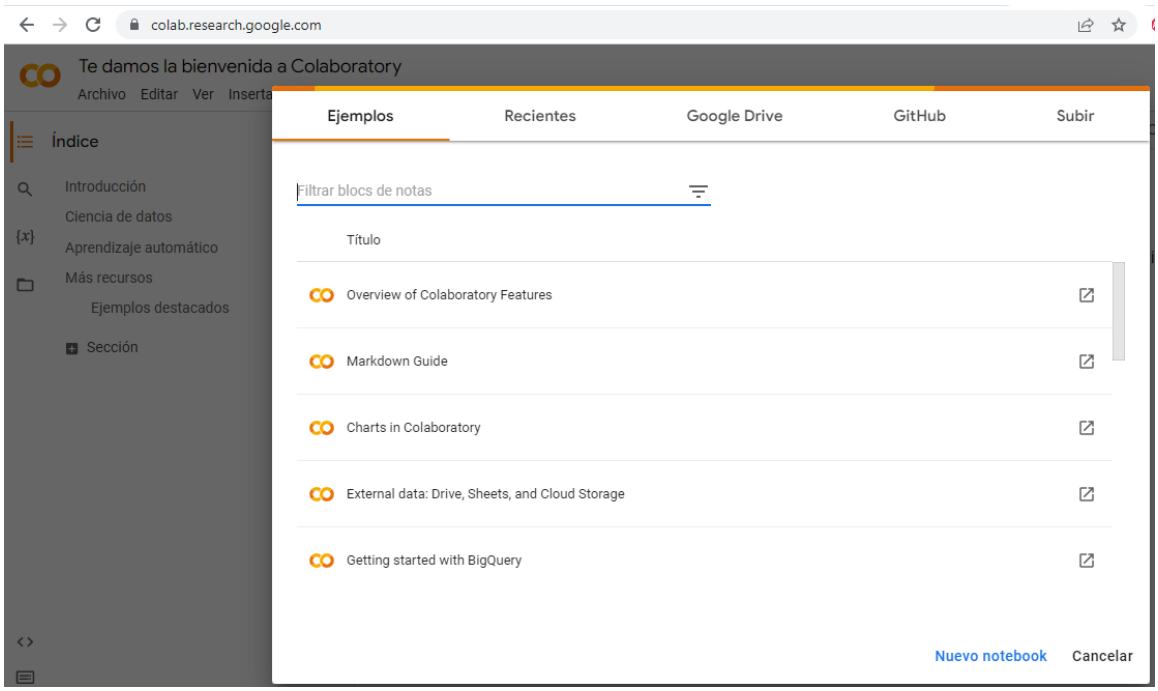


Figura 6.1: Página inicio Gooble Colab.
Fuente: autor

2. Seleccionar Google Drive, ahí es donde se guardara el cuaderno u código como indica la Figura 6.2

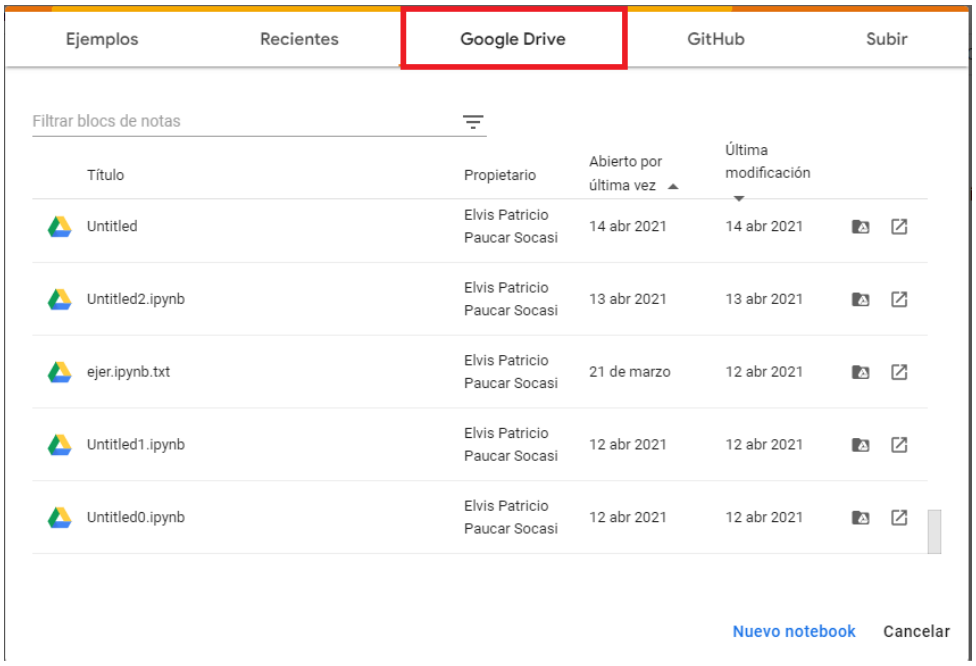


Figura 6.2: Selección Google Drive.
Fuente: autor

3. Para crear un nuevo cuaderno, hacer clic en la pestaña nuevo notebook en la parte inferior de la venta donde se desplegara un nuevo entorno como muestra la Figura 6.3.

Figura 6.3: Entorno Google Colab.
Fuente: autor

Antes de iniciar con la programación es importante verificar que el entorno de ejecución se encuentre seleccionado GPU, los siguientes pasos proporcionan un recorrido para cambiar la configuración del tiempo de ejecución de notebook: Dirigirse a entorno de ejecución, cambiar tipo de entorno de ejecución como indica la Figura 6.4

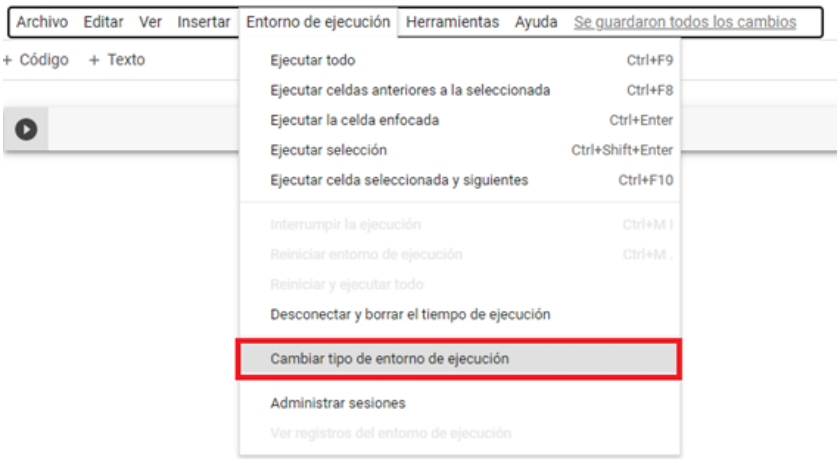


Figura 6.4: Cambio entorno de ejecución.
Fuente: autor

Aquí, existen opciones para cambiar el tiempo de ejecución de Python y el acelerador de hardware seleccionar GPU a continuación guardar, como indica la Figura 6.5

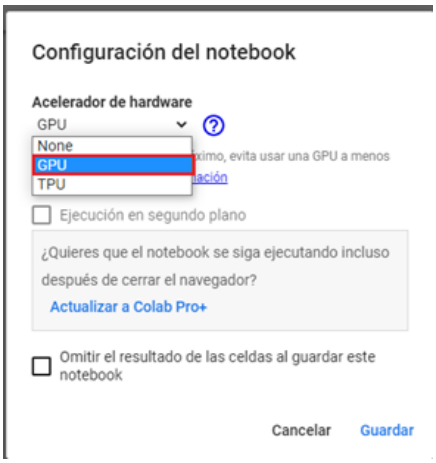


Figura 6.5: Cambio acelerador de hardware.
Fuente: autor

El motivo de realizar dicha configuración en el entorno de programación de Google Colab es la de aprovechar las tarjetas de video de los servidores de Google y de esta manera obtener más potencia computacional para el desarrollo de algoritmos mejorando la velocidad de procesamiento.

6.1.2 Enlace con Google Drive

La ventaja de realizar esta acción no es únicamente que podemos tener nuestros archivos almacenados en Google Drive y acceder fácilmente a ellos. Cuando se trabaja con machine learning o ciencia de datos, en la mayoría de casos contamos con archivos grandes de datos. Si cada vez que se conecta a un entorno distinto tenemos que subir estos archivos, perdiendo demasiado tiempo. Por ello, si se tiene estos archivos almacenados en Drive y montamos Drive en la máquina, podemos acceder a ellos como si estuvieran en local.

Para montar nuestro Drive en la máquina se escribe el siguiente código, como se muestra en la Figura 6.6.

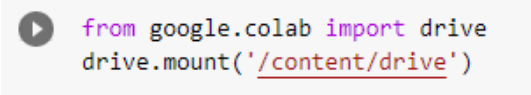


Figura 6.6: Código para enlazar con Drive.
Fuente: autor

Esto abre una URL donde que pide un código para autorizar a Colab a utilizar Drive y se tiene que pegar en el input que aparece, como se muestra en la Figura 6.7.

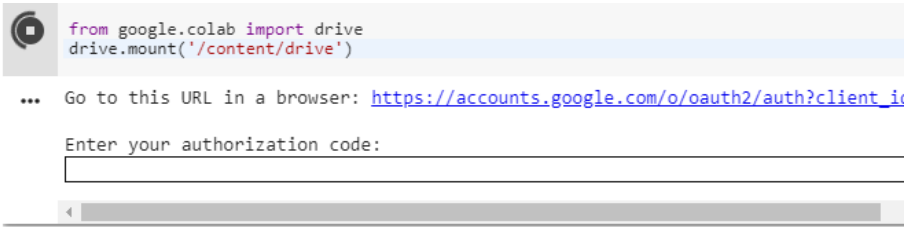


Figura 6.7: Montaje en Drive.
Fuente: autor

Una vez el proceso termina, nuestro Drive se ha montado dónde se ha indicado (/content/drive) y de esta forma se puede acceder a los archivos.

6.2 ANEXO II

6.2.1 Algoritmo para el entrenamiento del modelo de CNN

```
1
-  """
-  Algoritmo de Entrenamiento de Gestos
-  Programa realizado por: Elvis Paucar Socasi
5  """
-  !nvidia-smi
-
-  from google.colab import drive
-  drive.mount('/content/drive')
10
-  """# Cargamos librerias"""
-
-  import matplotlib.pyplot as plt
-  import numpy as np
15 import pandas as pd
-  import os
-  import fastai.basics as fai
-  import fastai.vision as fv
-  from fastai.metrics import error_rate, accuracy, ConfusionMatrix, fbeta
20 from fastai.vision import *
-  import torch
-  import torch.nn.functional as F
-  import torch.nn as nn
-  import torchvision.utils as utils
25 from pathlib import Path
-
-  """# Cargar DataFrame"""
-  path="/content/"
-  cargar_df=os.path.join(path,"/content/drive/MyDrive/Colab Notebooks
30 /base_de_datos_Gestos_Filtrados_30User.pkl")
-  print(cargar_df)
-
```

```
df_train ,df_test=pd.read_pickle(cargar_df)
df_train['set']='train'
df_test['set']='test'
df=pd.concat([df_train ,df_test],ignore_index=True,sort=False)
df
df.etiqueta.value_counts()
df.set.value_counts()

"""## Conectar DF a ANN"""

train_index=np.where(df.set=='train')[0]
print(train_index)

test_index=np.where(df.set=='test')[0]
print(test_index)

tecnicas=get_transforms(flip_vert=False,max_rotate=0,max_lighting=None)

data = (ImageList.from_df(df, path=path, cols='ubicacion')
.split_by_idxs(train_index, test_index)
.label_from_df(cols='etiqueta')
.transform(tecnicas, size=224)
.databunch(bs=6)
.normalize(imagenet_stats))

data.show_batch(rows=4,figsize=(5,5))

"""# Entrenamiento"""

ANN=cnn_learner(data,models.resnet34, metrics=[accuracy,error_rate,
FBeta(average='macro'), Precision(average='macro'),
Recall(average='macro')], callback_fns=ShowGraph)

ANN.fit_one_cycle(20)
ANN.recorder.plot_metrics()
ANN.recorder.plot_losses()

"""# Verificacion """
```

```
-
- matriz=ClassificationInterpretation.from_learner(ANN)
- matriz.plot_confusion_matrix(figsize=(4,4),dpi=100,cmap='ocean_r')
- matriz.plot_confusion_matrix(figsize=(4,4),dpi=100,normalize=True,
75 norm_dec=2,cmap='ocean_r')
- matriz.plot_top_losses(25,largest=True,figsize=(20,20))
- matriz.most_confused(1, 1)
-
- """# Exportacion de modelo """
80 ANN.export('/content/drive/MyDrive/Colab Notebooks/
- EntrenamientoDatosFiltrados/EntrenamientoGestosF_Resnet34.pkl')
-
- """# Modelo resnet50"""
-
85 ANN2=cnn_learner(data,models.resnet50, metrics=[accuracy,error_rate,
- FBeta(average='macro'), Precision(average='macro'),
- Recall(average='macro')], callback_fns=ShowGraph)
-
- ANN2.fit_one_cycle(20)
90 ANN2.recorder.plot_metrics()
- ANN2.recorder.plot_losses()
- matriz_2=ClassificationInterpretation.from_learner(ANN2)
- matriz_2.plot_confusion_matrix(figsize=(4,4),dpi=100,cmap='ocean_r')
- matriz_2.plot_confusion_matrix(figsize=(4,4),dpi=100,normalize=True,
95 norm_dec=2,cmap='ocean_r')
-
- matriz_2.plot_top_losses(25,largest=True,figsize=(20,20))
- matriz_2.most_confused(1, 1)
-
100 ANN2.export('/content/drive/MyDrive/Colab Notebooks/
- EntrenamientoDatosFiltrados/EntrenamientoGestosF_Resnet50.pkl')
-
- """# Modelo resnet101"""
-
105 ANN3=cnn_learner(data,models.resnet101, metrics=[accuracy,error_rate,
- FBeta(average='macro'), Precision(average='macro'),
- Recall(average='macro')], callback_fns=ShowGraph)
-
```

```
ANN3.fit_one_cycle(20)
ANN3.recorder.plot_metrics()
ANN3.recorder.plot_losses()
matriz_3=ClassificationInterpretation.from_learner(ANN3)
matriz_3.plot_confusion_matrix(figsize=(4,4),dpi=100,cmap='ocean_r')
matriz_3.plot_confusion_matrix(figsize=(4,4),dpi=100,normalize=True,
norm_dec=2,cmap='ocean_r')

matriz_3.plot_top_losses(25,largest=True,figsize=(20,20))
matriz_3.most_confused(1, 1)

ANN3.export('/content/drive/MyDrive/Colab Notebooks/
EntrenamientoDatosFiltrados/EntrenamientoGestosF_Resnet101.pkl')

"""# Modelo resnet152"""

ANN4=cnn_learner(data,models.resnet152, metrics=[accuracy,error_rate,
FBeta(average='macro'), Precision(average='macro'),
Recall(average='macro')], callback_fns=ShowGraph)

ANN4.fit_one_cycle(20)
ANN4.recorder.plot_metrics()
ANN4.recorder.plot_losses()
matriz_4=ClassificationInterpretation.from_learner(ANN4)
matriz_4.plot_confusion_matrix(figsize=(4,4),dpi=100,cmap='ocean_r')
matriz_4.plot_confusion_matrix(figsize=(4,4),dpi=100,normalize=True,
norm_dec=2,cmap='ocean_r')
matriz_4.plot_top_losses(25,largest=True,figsize=(20,20))
matriz_4.most_confused(1, 1)

ANN4.export('/content/drive/MyDrive/Colab Notebooks/
EntrenamientoDatosFiltrados/EntrenamientoGestosF_Resnet152.pkl')
```

6.3 ANEXO III

6.3.1 Algoritmo clasificación de gestos con modelo ResNet152

```
"""
Algoritmo de clasificación automatica de gestos utilizando señales EMG
Programa realizado por: Elvis Paucar Socasi
"""
"""Librerias necesarias"""
import fastai
from fastai.vision import *
import os
from pathlib import Path
import pandas as pd
import torch

""" Enlazar con Drive"""

from google.colab import drive
drive.mount('/content/drive')

""" Ubicación de modelo entrenado """

# Cargar el modelo entrenado desde un archivo .pkl
path = "/content/drive/MyDrive/Colab Notebooks/EntrenamientoDatos
Filtrados"
path_to_model = os.path.join(path,"EntrenamientoGestosF_Resnet152.pkl")
print(path_to_model)

"""Función para cargar modelo"""
learn = load_learner(path,path_to_model)

"""Imagen a clasificar"""
img = open_image(os.path.join("/content/drive/MyDrive/Colab Notebooks/
DatosGestosFiltrado/fist/img_55.jpg"))
img.show()
```

```
-
-  """Función para realizar clasificación de la imagen"""
35 # Utilizar el modelo para hacer predicciones en una imagen
-  pred_class,pred_idx,outputs = learn.predict(img)
-  print(pred_class)
```