Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Sistemas Computacionales

Detección y clasificación de objetos aplicado a un área de investigación: Reconocimiento de la Lengua de Señas Mexicana

Tesis para la obtención del grado de maestría en sistemas computacionales

Presenta: **Cristian Eli Jaime Gaytan**

Asesor: **MS. Victor Hugo Martinez Sanchez**

Tlaquepaque,Jalisco. Enero 2024

Agradecimientos

A través de estas líneas, deseo expresar mi más sincero agradecimiento a todas las personas que han contribuido de manera significativa a mi desarrollo, tanto en el ámbito profesional como en el personal.

En primer lugar, me gustaría expresar mi profundo agradecimiento a mis padres, cuyo inquebrantable apoyo ha sido fundamental en mi trayectoria educativa. Sin su dedicación y sacrificio, no habría sido posible llevar a cabo mis estudios universitarios. Reconozco el esfuerzo incansable que han realizado en pro de mi educación y su inquebrantable aliento para que nunca desista en mi búsqueda de conocimiento. A pesar de las adversidades y desafíos que hemos enfrentado a lo largo de los años, les estaré eternamente agradecido por su determinación y por estar siempre a mi lado cuando más los necesité.

Asimismo, deseo expresar mi gratitud al Instituto Tecnológico y de Estudios Superiores de Occidente por brindarme la oportunidad de cursar un posgrado. En particular, deseo destacar la invaluable contribución de mi asesor de proyecto de obtención de grado, el señor Víctor Hugo Martínez Sánchez. Su experiencia, conocimiento y constante motivación han sido fundamentales para el éxito de mi investigación.

No puedo pasar por alto agradecer a todos los profesores que compartieron sus conocimientos y sabiduría, y que guiaron mis pasos en el desarrollo de mi proyecto de investigación. Su dedicación y apoyo fueron cruciales para mi crecimiento académico, y estoy agradecido por el esfuerzo que realizaron para que sus estudiantes prosperen.

Por último, pero no menos importante, quiero expresar mi agradecimiento a mis amigos. A pesar de la distancia geográfica que algunos de nosotros enfrentamos, siempre hemos estado ahí el uno para el otro. Hemos compartido experiencias tanto en el ámbito académico como en el personal, y hemos brindado apoyo mutuo en los mejores y peores momentos de la vida. Su amistad ha sido un pilar fundamental en mi camino, y les estoy agradecido por su lealtad y compañía a lo largo de los años.

En resumen, quiero hacer patente mi reconocimiento a todas las personas que han sido parte integral de mi crecimiento y desarrollo, tanto en lo profesional como en lo personal. Sus contribuciones han sido invaluables, y estoy profundamente agradecido por cada uno de ustedes.

Dedicatoria

Quiero dedicar este trabajo final de maestría a todas las personas que han sido fundamentales en mi vida y en mi camino académico.

En primer lugar, quiero expresar mi profundo agradecimiento a Dios por otorgarme la oportunidad de vivir, por brindarme una familia maravillosa y unos amigos impresionantes. Cada uno de ustedes ha sido una bendición en mi vida, y estoy eternamente agradecido por su amor y apoyo incondicional.

A mis queridos padres, Martha Gaytán y Juan Antonio Jaime, les dedico este logro con todo mi corazón. Su apoyo emocional y económico ha sido fundamental en mi trayectoria académica, y su sacrificio ha hecho posible que alcance mis metas. Gracias por ser mis pilares inquebrantables y por creer en mí.

A mi hermana Vanessa, quiero expresar mi gratitud por estar a mi lado en cada paso de este viaje. A pesar de nuestras diferencias, sé que siempre puedo contar contigo, y tu apoyo incondicional ha sido un faro de luz en mi vida.

A mis apreciados amigos Juan Olvera, Guillermo Dávila, Juan Arredondo, Samara Escobar y Vladimir Rodríguez, les dedico este trabajo con profundo cariño. Su confianza en mí y su amistad han sido un tesoro invaluable a lo largo de los años. Hemos compartido momentos felices y tristes, y su presencia ha sido un consuelo constante. Siempre llevaré sus amistades en mi corazón, sin importar la distancia que nos separe.

A mis respetados maestros, quienes han compartido su conocimiento y dedicación conmigo, les agradezco de todo corazón. Nunca desistieron en enseñarme, incluso en las circunstancias más desafiantes, y por ello les estoy agradecido. Sus lecciones han sido cruciales en mi formación académica y personal.

Este logro no solo es mío, sino de todos ustedes que han estado a mi lado durante esta travesía. Gracias por ser parte de mi vida y por inspirarme a alcanzar este hito en mi formación. Este trabajo de obtención de grado lleva vuestros nombres en cada página, porque sin su apoyo y amor, no habría sido posible.

Resumen

El presente trabajo de maestría se enfoca en el estudio y análisis de la Lengua de Señas Mexicana (LSM), un lenguaje esencial para la comunidad sorda en México, que posee su propia sintaxis, gramática y léxico. La LSM se compone de una variedad de signos, cada uno con su propia lingüística y expresiones propias, lo que la convierte en una herramienta fundamental para que las personas sordas en México puedan expresar sus pensamientos y emociones, así como satisfacer sus necesidades comunicativas y cognitivas al interactuar con su entorno.

En la actualidad, se han desarrollado diversas propuestas y tecnologías con el propósito de facilitar la comunicación y comprensión de la LSM. Estas propuestas abarcan desde sistemas invasivos que se colocan en las manos y utilizan sensores de diversa índole, hasta sistemas menos invasivos que, mediante sensores externos, son capaces de captar el movimiento de las manos y las expresiones faciales de los usuarios.

A pesar de los avances en la tecnología y la investigación en este campo, es importante destacar que muchas investigaciones previas sobre el reconocimiento de la LSM han enfrentado desafíos en la implementación de señas debido a la complejidad de abordar todas las señas de cada región del país. En muchos casos, las investigaciones se han centrado en señas específicas, dividiéndolas en categorías estáticas y dinámicas, lo que ha limitado su aplicabilidad.

El objetivo principal de este proyecto de maestría es diseñar, implementar y verificar un modelo de redes neuronales que se base en modelos ya existentes, y que se enfoque en las señas estáticas de la Lengua de Señas Mexicana. Este enfoque permitirá el desarrollo de un sistema que pueda identificar y reconocer correctamente cada señal estática de la LSM, proporcionando información detallada sobre su significado y la forma adecuada de interpretarla. Este proyecto contribuirá de manera significativa a mejorar la accesibilidad y la comunicación de las personas sordas en México, facilitando la comprensión y el uso de la LSM en diversos contextos.

Índice

Índice de figuras

Lista de acrónimos y abreviaturas

AdaGrad	Adaptative Gradient Algorithm
AI	Artificial Intelligence
ANN	Artificial Neural Networks
ASL	American Sign Language
ASR	Automatic Speech Recognizer
CapsNet	Capsule Neural Network
CNN	Convolutional Neural Network
COCO	Common Objects in Context
DensNet	Densely Connected Convolutional Networks
DL	Deep Learning
DNN	Deep Neural Network
DTW	Dynamic Time Warping
ENS	Escuela Nacional de Sordomudos
FSL	France Sign Language
LSVRC	ImageNet Large Scale Visual Recognition Challenge
INEGI	Instituto Nacional de Estadística y Geografía
KNN	K-Nearest Neighbors
LSM	Lengua de Señas Mexicana
MLP	Multilayer Preceptron
MSE	Mean Square Error
OMS	Organización Mundial de la Salud
OPS	Organización Panamericana de la Salud
PASCAL	Pattern Analysis, Statistical Modelling, and Computational Learning
ReLU	Rectified Linear Unit
ResNet	Residual Network
RMSprop	Root Mean Square Propagation
SAR	Synthetic Aperture Radar
SASL	South Africa Sign Language
SGD	Stochastic Gradient Descent
SGDM	Stochastic Gradient Descent with Momentum
SSL	Spain Sign Language
SVM	Support Vector Machine
WRN	Wide Residual Network

1. INTRODUCCION

¿Qué ocurriría si elimin——áramos el canal principal a través del cual los seres humanos se comunican entre sí? La respuesta más evidente es que las personas no podrían expresar ideas, sentimientos o conceptos de manera efectiva. Este es el desafio central que enfrentan las personas con discapacidad auditiva. Sin embargo, gracias a la capacidad del ser humano para adaptarse a las adversidades, se han desarrollado lo que se conoce como lenguajes de señas, sistemas de comunicación basados en movimientos y expresiones que utilizan las manos, los ojos y gran parte del cuerpo.

Es importante destacar que, debido a la diversidad de idiomas en el mundo, e incluso dentro del idioma español, cada país ha desarrollado su propio lenguaje de señas. Esto ha dado lugar a diferencias significativas en la comunicación para las personas con discapacidad auditiva. En el contexto de México, la interprete e instructora Norma Montelongo señala que "existen diferentes señas dependiendo de los estados. No están unificadas las señas en México"[1].

Los datos proporcionados por el Instituto Nacional de Estadística y Geografía (INEGI) en el censo de 2020 son reveladores. Según el INEGI, el 16.5 % de la población mexicana tiene alguna discapacidad, y dentro de este grupo, el 24.4 % presenta problemas de audición, mientras que el 10.7 % tiene dificultades para comunicarse. Esto implica que aproximadamente 7,000,000 de personas en México enfrentan discapacidades relacionadas con la comunicación [2]. Estos números subrayan la importancia de abordar de manera efectiva las necesidades de comunicación de las personas con discapacidad auditiva.

En respuesta a esta realidad, se ha decidido enfocar el tema de investigación en el desarrollo de una aplicación diseñada para ayudar a las personas con discapacidad auditiva a comunicarse de manera más efectiva, centrándose en el reconocimiento de la Lengua de Señas Mexicana.

1.1. Antecedentes

Un asunto relevante que requiere consideración anticipada es la elección de la terminología apropiada para referirse a las personas con discapacidad, ya que algunas personas en la población aún pueden sentirse confundidas sobre cuál de los términos utilizar al hablar de este grupo de individuos. Según la definición de la Organización Panamericana de la Salud (OPS) en conjunto con la Organización Mundial de la Salud (OMS)[3], las personas con discapacidad se caracterizan por poseer deficiencias físicas, mentales, intelectuales o sensoriales de la larga duración que limitan su capacidad para participar plenamente y de manera efectiva en la sociedad, en igualdad de condiciones con otras personas.

De acuerdo con esta misma publicación, la OPS informa que aproximadamente el 12 % de la población en América Latina y el Caribe vive con al menos una forma de discapacidad, y estas personas a menudo enfrentan barreras significativas relacionadas con la salud, la comunicación, la educación y el acceso a oportunidades profesionales y financieras.

Cuando se aborda el tema de la discapacidad en México, es importante destacar que en 2016 el Senado de la república llevó a cabo un cambio significativo en la terminología, sustituyendo el término "capacidades diferentes" por la palabra "discapacidad"[4]. En la misma publicación citada, el entonces senador Jesús Casillas Romero argumento que el término "capacidades diferentes" era considerado arcaico, confuso y poco apropiado para describir la diversidad de experiencias y desafíos que enfrentan las personas con discapacidad.
La propuesta de modificar la terminología se fundamentó en estos argumentos y tuvo como objetivo principal la inclusión de las personas con discapacidad en todos los ámbitos de la sociedad. Este cambio de enfoque representa un paso importante hacia una comprensión más precisa y respetuosa de las experiencias de las personas con discapacidad en México.
Dado que el enfoque central de este proyecto se centra en la Lengua de Señas Mexicana (LSM), resulta esencial comprender sus orígenes. En 1867, se estableció la Escuela Nacional de Sordomudos (ENS) en la Ciudad de México, con el propósito de ensenar a los alumnos a utilizar las señas existentes en ese periodo, promoviendo así la creación de vínculos sociales dentro de la comunidad sorda. Vale la pena destacar que Eduardo Huet fue el fundador de la ENS y desempeño un papel fundamental en su desarrollo. Debido a la influencia de sus orígenes franceses, la LSM actual guarda similitudes con la Lengua de Señas Francesa (French Sign Language o FSL)[5]. Este vínculo histórico entre la LSM y la Lengua de Señas Francesa ha influido en la evolución y estructura de la LSM tal como la conocemos en la actualidad.

1.2. Justificación

La comunicación oral representa la forma principal de interacción entre los seres humanos. Sin embargo, es importante reconocer que existen individuos que, ya que sea desde su nacimiento o debido a circunstancias adquiridas, enfrentan una discapacidad auditiva que les impide comunicarse de manera oral. Es en este contexto que surge la Lengua de Señas Mexicana (LSM) como una herramienta crucial para permitir a estas personas expresar sus pensamientos, sentimientos e ideas a quienes les rodean.
Uno de los desafíos fundamentales asociados con la LSM radica en que, al igual que en muchos otros países, existen variaciones regionales en esta lengua. Sin embargo, se han establecido señas estandarizadas para letras del abecedario, lo que facilita el proceso de aprendizaje básico de la LSM.
En el ámbito de la investigación sobre lenguas de señas, ya sea la LSM mexicana u otras, los expertos han identificado dos categorías principales de señas: las señas dinámicas y las señas estáticas. En general, se considera que trabajar con señas dinámicas presenta ventajas significativas, dado que estas no involucran movimiento. Sin embargo, la implementación de señas dinámicas es más compleja, ya que requiere el uso de software y/o hardware especializado para detectar movimientos de manos, brazos y expresiones faciales.

Se han desarrollado dos tipos de prototipos para abordar esta cuestión: los prototipos invasivos, que se conectan directamente a las manos y detectan su movimiento, y los prototipos no invasivos, que utilizan sensores de profundidad o dispositivos de captura de imágenes y software para interpretar las señales de los sensores.
En el marco de este proyecto, se propone el desarrollo de una aplicación que se enfoque exclusivamente en detectar los gestos estáticos de la LSM, dado que la implementación de señas dinámicas requeriría un esfuerzo considerable y más tiempo.
Finalmente, es importante resaltar que el objetivo central de este proyecto va más allá de mejorar la comunicación de las personas con discapacidad auditiva. También se busca facilitar la comunicación entre personas que no tienen esta discapacidad y aquellas que, si la tienen, promoviendo así la inclusión y la igualdad en la sociedad. El proyecto aspira a fomentar un entorno en el que todas las personas puedan interactuar y comunicarse de manera efectiva, independientemente de sus capacidades auditivas

1.3. Problema

El problema central abordado en este proyecto se relaciona con los desafíos del aprendizaje de la Lengua de Señas Mexicana (LSM) en diversas comunidades y, por consiguiente, las limitaciones en la comunicación que surgen debido a la falta de dominio de este lenguaje. Esta problemática se origina en la insuficiente o prácticamente inexistente enseñanza de la LSM en las escuelas de todos los niveles educativos. Como resultado, el conocimiento y uso de la LSM por parte de las personas no discapacitadas es limitado. En respuesta a esta situación, el gobierno mexicano y los gobiernos estatales han lanzado campañas que ofrecen cursos gratuitos para aquellos interesados en aprender la LSM. Sin embargo, estas iniciativas han mostrado ser poco eficaces en su implementación y alcance. A pesar de los obstáculos, existen individuos y profesionales que han persistido en su compromiso de integrar a las personas con discapacidad auditiva en la sociedad, en especial en el ámbito educativo.
Un ejemplo destacado de esto es el lingüista y educador Gustavo Escoba, quien aborda el desafio fundamental de que nuestra lengua es predominantemente oral y las personas sordas carecen del canal auditivo necesario para recibir esta forma de comunicación [6] Además, señala que aquellos que nacen en familias con discapacidad auditiva adquieren la LSM como lengua materna, lo que les permite comunicarse dentro de su comunidad, pero enfrentan dificultades en la interacción con personas que no dominan este lenguaje. A lo largo de su trayectoria, el maestro Gustavo ha logrado una notable inclusión tanto académicamente como laboral de una parte significativa de los 2.3 millones de mexicanos que enfrentan una discapacidad auditiva. Su dedicación se traduce en un impacto positivo y tangible en la vida estas personas. En una entrevista documentada en [6], el maestro Gustavo menciona que su investigación ha culminado en la creación de tres libros de gramática de la Lengua de Señas Mexicana (LSM), los cuales han sido instrumentales en el proceso de aprendizaje e inclusión de las personas con discapacidad auditiva.

En cuanto al enfoque de este proyecto, se centra en la detección de señas estáticas de la LSM mediante el uso de una aplicación. Esta elección se basa en la consideración de que las señas estáticas son más frecuentes que las señas dinámicas, y su detección es más eficiente en términos de tiempo y recursos. Además, se ha observado que las señas dinámicas representan letras que utilizan con menor frecuencia en el idioma español, como se ilustra en el video [7] presentado por la BBC. Por tanto, el enfoque en las señas estáticas es un paso inicial estratégico en la búsqueda de mejorar la comunicación y la inclusión de las personas con discapacidad auditiva en a la sociedad.

1.4. Objetivos

En el proceso de elaboración de este proyecto, se ha formulado un objetivo principal que estará respaldado por cinco objetivos específicos. Estos objetivos particulares serán detallados y desarrollados en las secciones subsiguientes para brindar una comprensión exhaustiva de la dirección y los alcances de la investigación.

1.4.1. Objetivo General

El objetivo general de este proyecto consiste en diseñar, implementar y verificar uno o varios modelos de redes neuronales basados en modelos previamente existente de redes neuronales, con el propósito de lograr el reconocimiento preciso de la Lengua de Señas Mexicana.

1.4.2. Objetivos Específicos

Los objetivos particulares que se han establecido para este proyecto son los siguientes:

- Obtener uno o varios modelos de redes neuronales que sean adecuados para los propósitos de la investigación.
- Recopilar un conjunto de datos que contenga imágenes relacionadas con la Lengua de Señas Mexicana.
- Identificar y poner en práctica una o varias arquitecturas de redes neuronales que se ajusten a los objetivos específicos del proyecto.
- Evaluar y validar el conjunto de datos frente a los modelos de redes neuronales propuestos en el contexto del proyecto.
- Determinar la eficacia de los modelos de redes neuronales mediante una evaluación exhaustiva utilizando el conjunto de datos previamente establecido.

1.5. Innovación tecnológica

La innovación tecnológica primordial en este proyecto se centra en la capacidad de reconocer los gestos realizados en imágenes mediante un dispositivo electronico, incluyendo la capacidad de predecir la letra de la señalización realizada y anticipar la posición de los dedos principales en dicha seña. Este reconocimiento de gestos se logra a través de unos de los enfoques más vanguardistas y destacados de los últimos años, conocido como redes neuronales convolucionales, utilizando puntos clave como parte integral del proceso.

2. ESTADO DEL ARTE

2.1. Clasificación de Imágenes

La visión computacional se enfoca en la resolución de problemas relacionados con la clasificación de imágenes, la localización y la detección de objetos en diversos entornos [8]. Una de las técnicas principales empleadas para la clasificación de imágenes es el aprendizaje automático o machine learning (ML), en particular, la implementación de redes neuronales [9]. La función fundamental del ML es permitir que un sistema aprenda de datos diversos mediante técnicas de programación específicas. Sin embargo, este proceso no es trivial, ya que el algoritmo requiere datos de entrenamiento adecuados. Dependiendo del algoritmo y los datos de entrenamiento utilizados, el modelo de machine learning generara información relacionada con el proceso de entrenamiento [10].
Existen diversas técnicas que contribuyen a mejorar la precisión de los modelos de ML, y la elección de enfoque depende del problema a resolver, así como del tipo y el volumen de datos involucrados. Algunas aplicaciones de estas técnicas incluyen la organización automática de imágenes, el manejo de grandes bases de datos visuales y el reconocimiento de imágenes, entre otros. En estos contextos, se busca lograr la clasificación más precisa posible [11].
En el ámbito de clasificación de imágenes, se puede identificar la siguiente estructura, como se menciona en[11]:

1. **Preprocesamiento de Imagen:** En esta etapa se realizan modificaciones y ajustes que incluyen cambios en el tamaño de la imagen, ajustes en las escalas de colores, conversión a escalas de grises, aplicación de efectos de difuminación, entre otros procesos.
2. **Detección de Objetos:** En esta fase se lleva a cabo la identificación y clasificación del objeto de interés mediante la segmentación de la imagen.
3. **Características y entrenamiento:** En este paso, se aplican métodos estadísticos o técnicas de Deep Learning (DL) para identificar patrones de interés en una imagen. Estos patrones son fundamentales para que el modelo pueda diferenciar entre diferentes objetos y categorías. El proceso implica que el modelo aprenda a partir de una base de datos establecida, y se conoce como el entrenamiento del modelo.
4. **Clasificación del objeto:** En esta última etapa, los objetos se categorizan en clases predefinidas utilizando técnicas de comparación de patrones y la búsqueda de patrones previamente establecidos.

2.1.1. Técnicas de Clasificación de Imágenes

En el ámbito de la clasificación de imágenes, se emplean diversas técnicas destacadas, entre las cuales se incluyen los siguientes clasificadores: Máquinas de Vectores de Soporte (Support Vector Machine o SVM), Árboles de Decisiones, K vecinos más Próximos (k-Nearest Neighbors o KNN), Redes Neuronales Artificiales (Artificial Neural Networks o ANN) y, por último, Redes Neuronales Convolucionales (Convolutional Neural Networks o CNN)[12]. Cada una de estas técnicas presenta sus propias características, ventajas y aplicaciones específicas en el ámbito de la clasificación de imágenes, lo que permite a los investigadores y profesionales seleccionar el enfoque más adecuado según los requisitos y las particularidades de sus proyectos.

2.1.1.1 Máquinas de Vectores de Soporte

El método de clasificación de imágenes SVM tuvo sus orígenes en el campo de la ciencia computacional a finales de la década de 1990. En su concepción inicial, este método fue desarrollado principalmente para la clasificación binaria, pero con el transcurso del tiempo, su aplicabilidad se expandió hacia la resolución de problemas que requerían clasificación múltiple y regresión[13].
La versatilidad de SVM en la clasificación y regresión de datos permitió su adaptación a diversas áreas de aplicación. Entre las áreas destacadas en las que se ha implementado con éxito se incluyen la clasificación de texto, el procesamiento de señales de radar y la clasificación de imágenes. Además, ha encontrado aplicaciones en campos como el análisis de riesgo crediticio, diagnósticos médicos y extracción de información [14].
En el contexto de la clasificación de textos, los SVM han desempeñado un papel fundamental a lo largo de los años, contribuyendo a la detección de correos electrónicos no deseados (spam), facilitando la detección de idiomas en diversos traductores, como Google Translate, y desempeñando un papel importante en las numerosas asistentes virtuales disponibles en el mercado, como Siri o Alexa[15].
Además, algunos estudios han aplicado la técnica de SVM en el ámbito de la teledetección, particularmente en el procesamiento de datos de radar de Apertura Sintética (Synthetic-aperture radar o SAR). Estos estudios han abordado tareas que van desde la estimación de parámetros que separan datos en diferentes clases hasta la clasificación de imágenes multiespectrales y elementos multifrecuencia [16], [17]. Estas aplicaciones ilustran la versatilidad y la efectividad de los SVM en una amplia gama de dominios y problemáticas.

2.1.1.2 Árboles de Decisiones

Los árboles de decisiones, como se mencionó en [18], fueron propuestos por Leo Breiman, Jerome Friedman, Richard Olshen y Charles Stone en 1984, con un enfoque específico en el ámbito estadístico. A lo largo de los años, los árboles de decisiones se han aplicado a dos tipos principales de problemas: aquellos relacionados con la regresión y aquellos que se detallan en [19], centrados en la clasificación.
Una de las implementaciones más significativas de los árboles de decisiones se encuentra en el ámbito del marketing moderno. Conforme se destaca en [20], las empresas han adoptado esta técnica debido a su capacidad para mejorar la precisión en las campañas de productos próximos a ser lanzados al mercado. Los árboles de decisiones permiten evaluar el rendimiento de estos productos, así como de productos similares ofrecidos por competidores. Además, facilitan la segmentación del público objetivo y la optimización de la producción de anuncios dirigidos a objetivos específicos.

2.1.1.3 K Vecinos más Próximos

En 1951, los renombrados matemáticos y estadísticos Evelyn Fix y Joseph Lawson Hodges Jr., en colaboración con la Escuela de Medicina Aeronáutica de la Fuerza Aérea de los Estados Unidos, desarrollaron un método de clasificación no paramétrica. Este método se enfocó en la necesidad de realizar estimaciones cuando las probabilidades son desconocidas o difíciles de determinar, como se menciona en [21]-[23].
En 1967, Thomas Cover y Peter Hart demostraron que el error de clasificación de este método está acotado al doble de la tasa de error de Bayes, lo que lo convirtió en una característica fundamental. Esto sentó las bases para futuros estudios relacionados, como se detalla en los siguientes artículos [21]–[23].
Hoy en día, gracias a las contribuciones de científicos que han colaborado en el desarrollo del método de los K Vecinos más Próximos (NN), este enfoque de clasificación se ha implementado en diversas aplicaciones. Estas aplicaciones incluyen el reconocimiento de patrones, la detección de intrusos [24], sistemas de recomendación y detección de anomalías [25]. Sin embargo, la implementación más destacada del método KNN se encuentra en el campo de la minería de datos (data mining).
Es importante destacar que, aunque en esta última aplicación los costos computacionales son más elevados en comparación con otros algoritmos utilizados, el algoritmo KNN sigue siendo la elección preferida. Esto se debe a que, si bien las predicciones no se realizan con una frecuencia recurrente, se requiere un alto nivel de precisión [26], [27]. La capacidad de KNN para brindar resultados precisos en tareas de minería de datos lo convierte en una herramienta valiosa para la toma de decisiones basada en datos en una amplia gama de aplicaciones.

2.1.1.4 Redes Neuronales Artificiales

Uno de los primeros algoritmos predecesores de las redes neuronales fue creado en 1958 por el psicólogo estadounidense Frank Rosenblatt y se denominó perceptrón. Este perceptrón recibía entradas binarias variables y producía una sola salida binaria, cuyo valor dependía de la multiplicación de los pesos asignados [28].
En 1965, como resultado de los avances en los años siguientes, surgió lo que se conoce como perceptrón multicapa, que contenía más neuronas en su composición. Este avance introdujo conceptos clave como la capa de entrada, la capa oculta y la capa de salida, todas con valores binarios. Sin embargo, los pesos asignados aún se ajustaban manualmente [28], [29]. El perceptrón multicapa fue utilizado por algunos bancos para identificar riesgos en los créditos, tanto positivos como negativos, ayudando a identificar a los clientes que representaban algún riesgo [30].
Sin embargo, en la década de los 80, se produjo un auge en las innovaciones en las redes neuronales, y estas innovaciones aún se implementan en la actualidad. Dado que uno de los principales desafíos en ese momento era lograr que las neuronas aprendieran de manera automática sin requerir que los investigadores ingresaran manualmente los datos, se introdujeron las neuronas sigmoides. Estas neuronas actúan como una función .ªplastantez generan una salida en su rango de 0 a 1, proporcionando valores reales en su salida [31].
En 1986, se popularizó un algoritmo conocido como backpropagation, que ha sido ampliamente utilizado debido a su eficiencia en el entrenamiento de redes neuronales. Este algoritmo se basa en el método çhain rulez ha permitido que expertos en redes neuronales entrenen con éxito redes multicapa. Además, los nodos de estas redes aprendieron características similares a las que los expertos humanos diseñarían. Gracias a su eficacia, el método de backpropagation se ha implementado en diversas áreas, mejorando tanto los tiempos como los costos de implementación [32], [33].

2.1.1.5 Redes Neuronales Convolucionales

A finales de la década de los 80, específicamente en 1989, el informático estadounidense Yann LeCun comenzó a desarrollar la idea de las Redes Neuronales Convolucionales (CNN). En 1998, publicó un trabajo titulado "Gradient-Based Learning Applied to Document Recognition" en el que se presentaba un diseño que él mismo creó, denominado LeNet, una CNN simple que consta de 7 capas con parámetros ajustables [28], [34], [35].
Con el paso del tiempo y el desarrollo de nuevas arquitecturas basadas en CNN, surgió ImageNet, un conjunto de datos de imágenes. ImageNet creó un desafío denominado "Desafío de reconocimiento visual a gran escala de ImageNet (ILSVRC)" con el objetivo de evaluar algoritmos para la detección de objetos y la clasificación de imágenes a gran escala [36], [37]. Esta competencia comenzó en 2010 y llevó a los investigadores a colaborar para comparar modelos de machine learning y visión por computadora, con el objetivo de mejorar su rendimiento en la clasificación de imágenes.

Gracias al desafío ILSVRC, en 2012 Alex Krizhevsky y Geoffrey Hinton crearon AlexNet, la primera arquitectura CNN en ganar el mencionado desafío. AlexNet redujo la tasa de error del 25.8 % al 16.4 %. La diferencia principal respecto a LeNet fue que AlexNet utilizó 8 capas, implementó unidades lineales rectificadas (ReLU) en lugar de la función tangente hiperbólica (tanh) y aprovechó múltiples GPUs para acelerar el entrenamiento [38].
En el año siguiente, Zeiler y Fergus crearon ZFNet, similar a AlexNet, pero con algunas modificaciones en las capas y un aumento en el número de filtros. Además, introdujeron conceptos como la visualización profunda de funciones, la invariancia y la importancia de estos conceptos [39].
En 2014, el Visual Geometry Group de la Universidad de Oxford desarrolló la arquitectura VGGNet, que consideraba la profundidad como un componente esencial en el diseño, junto con una arquitectura uniforme y campos receptivos reducidos [40].
En ese mismo año, Google presentó su propia metodología llamada GoogleNet, que se centraba en redes profundas con el objetivo de mejorar la eficacia en el conteo de parámetros, el ahorro de memoria y el proceso de cálculo. Esta arquitectura utilizó 22 capas sin ninguna completamente conectadas, lo que redujo significativamente el tamaño en comparación con AlexNet y VGG, hasta 12 veces menos que AlexNet y 28 veces menos que VGG [41].
Finalmente, en 2015, Kaiming He introdujo la Red Neuronal Residual (ResNet), que presentaba conexiones residuales y normalización por lotes pesados. ResNet constaba de un total de 152 capas y logró reducir el error de la arquitectura de VGG del 7.3 % al 3.6 %, superando el rendimiento humano en este conjunto de datos [42], [43].

2.2. Lengua de Señas Mexicana

En 1867 se fundaría la Escuela Nacional de Sordomudos (ENS) en la Ciudad de México, donde se enseñaba a los alumnos a utilizar las señas existentes en ese tiempo para que las personas crearan vínculos sociales entre la misma comunidad sordomuda. Desde la creación de la escuela, se crearon dos escuelas simultáneas: una dedicada para niños y adolescentes y la escuela normal que enseñaba tanto la teoría como la práctica para que estas personas que tomaban clases en la segunda se volvieran profesores de sordomudos [44].
Sin embargo, a mediados del siglo XX, como se menciona en [45], la ENS cerraría sus actividades por motivos desconocidos, pero se atribuye el cierre a la nueva política de ese momento en la cual el estado favorecía la educación de las personas sordas a través de la utilización de la lengua oral.
A inicios del siglo XX, gracias al modelo educativo implantado por la Secretaría de Educación y la Dirección General de Educación Especial, todos los maestros y especialistas de la LSM en ese tiempo asumieron que el enfoque de aprender español para alguien con discapacidad auditiva, debido a que se asumía que aprender de manera oral y escrito del español, ayudaba a la integración de las personas a una escuela regular [45].

A comienzos de los 80's, las escuelas de educación especial empezaron a implementar una filosofía llamada "comunicación total", la cual tiene el enfoque orientado a desarrollar la comunicación de manera efectiva y equitativa a través de utilizar todos los medios disponibles de las personas, con el fin de comprender y ser comprendido, mediante la utilización de objetos, imágenes, símbolos, signos y a su vez que las personas utilicen lo antes mencionado como medio de comunicación [46].
Sin embargo, no fue hasta 2003 que la Lengua de Señas Mexicana (LSM) se reconoció oficialmente como una lengua nacional en México, junto con otras lenguas indígenas y el español. Este reconocimiento facilitó el aprendizaje de la LSM, ya que décadas antes se había enfocado en el uso de la voz y la lectura de labios [47]. Más tarde, el 10 de junio de 2005, se estableció el Día Nacional de la Lengua de Señas Mexicana, reconociendo a esta lengua como patrimonio lingüístico de México [48].
La LSM tiene su propia estructura y gramática basada en el español que se habla en México [49]. Esta lengua comprende más de 1000 señas utilizadas en todo el país y consta de un vocabulario y gramática complejos que varían según la comunidad [50]. Sin embargo, se busca simplificar la LSM a 27 señas en total, de las cuales 21 son estáticas y 6 son dinámicas, expresadas mediante movimientos de las manos. Estas señas se realizan utilizando posturas y movimientos de los dedos, las manos, los brazos e incluso los puños y otras partes del cuerpo, además de expresiones faciales [49], [50].

2.3. Sistemas de reconocimiento

El reconocimiento de la Lengua de Señas Mexicana (LSM) se basa principalmente en la implementación de sistemas electrónicos y dispositivos que capturan la posición, movimiento y aceleración de los dedos, manos y brazos, lo que a menudo resulta en sistemas invasivos para las personas.
Como se menciona en el documento ***"Mexican Sign Language Recognition Using Kinect and Data Time Warping Algorithm"*** [51], los autores utilizaron un sensor Microsoft Kinect para reconocer en tiempo real los signos de la LSM y transformarlos en voz y texto. Los componentes clave de hardware utilizados por este dispositivo son el sensor infrarrojo y el sensor de profundidad. El sensor infrarrojo emite rayos infrarrojos, mientras que el sensor de profundidad lee los rayos reflejados para determinar la distancia entre el objeto y el sensor [52]. En el documento [51], se empleó el dispositivo Kinect junto con un algoritmo de deformación dinámica del tiempo (DTW) para interpretar los gestos realizados con las manos, siendo DTW un método que mide la similitud temporal entre dos secuencias temporales.
Por otro lado, un documento similar al mencionado previamente es el realizado por Omar Santiago, Morales Caballero y Felipe Trujillo en [53]. En este caso, también se utilizó el dispositivo Kinect, pero se implementó un sistema de traducción de lenguaje de voz a lenguaje de señas mediante modelos 3D y un Reconocedor Automático de Voz (ASR).
Por último, se han implementado otros sistemas de reconocimiento que utilizan cámaras para seguir el movimiento de las manos, como se describe en el documento [54]. Este enfoque se complementó con la aplicación de un Modelo Oculto de Márkov en tiempo real y se implementó en la Lengua de Señas Americana (ASL).

3. MARCO TEÓRICO

3.1. Inteligencia Artificial

La Inteligencia Artificial, también conocida como Artificial Intelligence (AI), se refiere a la capacidad de una computadora digital o de un robot controlado por una entidad para llevar a cabo tareas relacionadas con la inteligencia humana. Este término generalmente se utiliza para el desarrollo de sistemas que poseen procesos intelectuales característicos de los seres humanos, como el razonamiento, el aprendizaje basado en experiencias y la capacidad de descubrir significados, entre otros [55].
Además, el científico y matemático estadounidense John McCarthy menciona en su trabajo ***What is Artificial Intelligence?*** que la IA es la ciencia y la ingeniería dedicada a la creación de máquinas inteligentes programadas y diseñadas por expertos en informática. La IA se enfoca en utilizar dispositivos electrónicos, como las computadoras, para comprender la inteligencia humana, aunque no se limita necesariamente a métodos biológicamente observables [56].

3.1.1. Aprendizaje Automático Aplicado

El aprendizaje automático, también conocido como **machine learning**, es una rama de la inteligencia artificial que permite a las computadoras aprender sin ser programadas explícitamente para ello. Esta tecnología se utiliza principalmente para el reconocimiento de patrones y la realización de predicciones basadas en un conjunto de datos previamente seleccionado, y se aplica en diversas áreas, como servicios de streaming y respuestas automáticas en correos electrónicos [57].
Los algoritmos de aprendizaje automático son conjuntos finitos de operaciones que permiten encontrar soluciones a problemas específicos [58]. Estos algoritmos se dividen en tres categorías principales, según se describe en [59]:

- **Aprendizaje Supervisado:** En este enfoque, se utiliza un conjunto de datos previamente etiquetados para entrenar al algoritmo. A medida que se ingresan datos adicionales en el modelo desarrollado, este ajusta sus valores internos para mejorar la precisión. El aprendizaje supervisado se divide en dos tipos: clasificación, que asigna datos de prueba a categorías predefinidas mediante etiquetas, y regresión, que se utiliza para comprender la relación entre variables y realizar proyecciones [60].
- **Aprendizaje No Supervisado:** En el aprendizaje no supervisado, los algoritmos analizan y agrupan conjuntos de datos que no están etiquetados previamente. Estos algoritmos pueden descubrir patrones ocultos y similitudes en los datos sin la intervención humana, lo que los hace útiles para la exploración y segmentación de información [61].

- **Aprendizaje por Refuerzo:** El aprendizaje por refuerzo, como se describe en [62], es un método de entrenamiento de machine learning en el que un agente toma decisiones a través de la prueba y el error. El agente recibe recompensas por lograr comportamientos deseados y es castigado por comportamientos no deseados. Este enfoque es adecuado para situaciones en las que el sistema debe aprender a tomar decisiones basadas en experiencias previas y resultados.

Estas categorías de algoritmos de aprendizaje automático se utilizan en diversas aplicaciones y juegan un papel fundamental en la automatización de tareas y la toma de decisiones basadas en datos

3.1.2. Aprendizaje Profundo

El aprendizaje profundo, también conocido como Deep Learning (DL), se define según IBM en [63] como una subdisciplina del aprendizaje automático. Esta definición se complementa con la aportación de Jason Brownlee en [64], quien señala que el DL se inspira en la estructura y el funcionamiento del cerebro, a menudo denominados Redes Neuronales Artificiales (ANN, por sus siglas en inglés). Siguiendo la descripción de IBM, el DL introduce avances significativos en el procesamiento de datos en comparación con el aprendizaje automático convencional.
En primer lugar, el DL tiene la capacidad de abordar datos no estructurados, como texto e imágenes, lo que amplía su aplicabilidad a una variedad de dominios. Además, se destaca por automatizar la extracción de características, eliminando en gran medida la necesidad de intervención manual en este proceso. Esto implica que los algoritmos de DL pueden discernir de manera autónoma las características más relevantes para distinguir variables entre sí [63]. Este enfoque revolucionario ha demostrado un gran potencial en diversas aplicaciones de la inteligencia artificial.

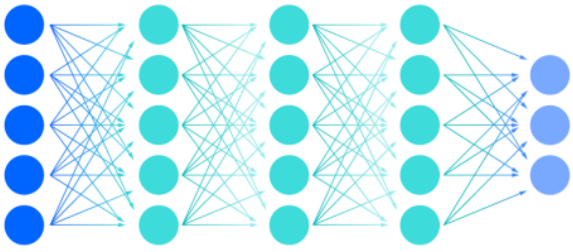


Figura 1: Arquitectura básica de una red neuronal profunda [65]

3.2. Redes Neuronales

Las redes neuronales artificiales, también conocidas como Artificial Neural Networks (ANN), representan un subconjunto fundamental del campo del aprendizaje automático, según la definición proporcionada por IBM [66]. Estas redes tienen la capacidad de reconocer patrones y abordar una amplia gama de problemas en el ámbito de la Inteligencia Artificial (IA), así como en el aprendizaje automático y aprendizaje profundo.
La arquitectura de una ANN se compone de capas de nodos interconectados, que incluyen una capa de entrada, una o varias capas ocultas y una capa de salida. Cada uno de estos nodos, también denominados neuronas artificiales, establece conexiones con otros nodos, asignando pesos y umbrales a cada conexión. Cuando la salida de un nodo supera el umbral especificado, dicho nodo se activa y transmite los datos correspondientes a la siguiente capa.
Conforme se destaca en el informe de IBM [65], las redes neuronales aprenden a partir de datos de entrenamiento y perfeccionan su precisión con el tiempo. Una vez que estos algoritmos de aprendizaje se han afinado, se convierten en herramientas sumamente poderosas para la informática y la inteligencia artificial. Este hecho permite llevar a cabo tareas como el reconocimiento de voz o de imágenes en cuestión de minutos, en comparación con las horas que requeriría la identificación manual por parte de un experto humano. Este enfoque agiliza significativamente la clasificación y el agrupamiento de información a alta velocidad.

3.2.1. Perceptrón

El perceptrón, una de las redes neuronales más antiguas según se documenta en [65], fue concebido por Frank Rosenblatt en el año 1958. Este tipo de red neuronal se caracteriza por su simplicidad, ya que consiste en una sola neurona, lo que lo convierte en el tipo más básico de red neuronal disponible. Los perceptrones se utilizan para realizar cálculos con el propósito de identificar características o patrones en los datos de entrada.

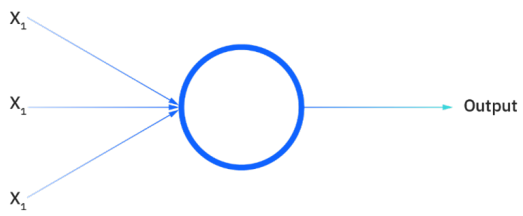


Figura 2: Diseño básico de un Perceptrón [65]

En esencia, los perceptrones representan un algoritmo de aprendizaje supervisado diseñado principalmente para realizar clasificaciones binarias. Este enfoque busca permitir que las neuronas involucradas en el perceptrón aprendan y procesen elementos a partir de conjuntos de datos de entrada específicos.
Los perceptrones desempeñan un papel fundamental en proyectos relacionados con el aprendizaje automático. Principalmente, se utilizan como clasificadores de datos, algoritmos de simplificación y como controladores de capacidad en algoritmos de clasificación binaria. Su implementación se basa en el aprendizaje supervisado, lo que implica la necesidad de enseñar a los algoritmos a realizar predicciones utilizando datos previamente etiquetados, según se detalla en [67].
Siguiendo las reglas del aprendizaje del perceptrón, conocidas como Perceptron Learning Rule, un algoritmo tiene la capacidad de aprender de manera automática los factores de ponderación óptimos. En este proceso, las características de los datos de entrada se multiplican por los pesos asignados a cada una de ellas. Dado que un perceptrón recibe múltiples señales de entrada, si la suma de estas señales supera un umbral predeterminado, se genera una señal que permite que la información.

3.2.2. Perceptrón Multicapa

En el contexto de las Redes Neuronales Artificiales (ANN), es importante mencionar los perceptrones multicapa, conocidos en inglés como Multilayer Perceptron (MLP). Estas redes neuronales son una categoría específica que consta de al menos tres nodos, y cada neurona, a excepción de la capa de entrada, emplea una función de activación no lineal [68].
El perceptrón multicapa se compone de tres capas fundamentales: la capa de entrada, una o varias capas ocultas en el medio y la capa de salida. Se caracteriza por tener conexiones no superpuestas, pero interconectadas, de manera que la salida de una neurona se convierte en la entrada de la siguiente, como se describe en [69].
En el funcionamiento de un perceptrón multicapa, se pueden distinguir dos etapas cruciales [69]:

- **Aprendizaje No Supervisado:** En esta etapa, la red calcula la salida a partir de los valores de entrada. Los datos fluyen desde la capa de entrada a través de las capas ocultas hasta llegar a la capa de salida, generando así una predicción.
- **Aprendizaje:** En esta fase, conocida como retropropagación del error (backpropagation), se ajustan los pesos de las conexiones para minimizar la diferencia entre el valor estimado por la red y el valor verdadero deseado. Este proceso se basa en la función gradiente de error y tiene como objetivo mejorar la precisión de la red mediante la modificación de los pesos.

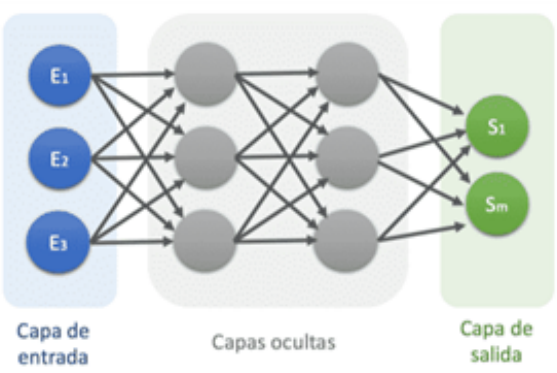


Figura 3: Diseño básico de un perceptrón multicapa [69]

Finalmente, es relevante descubrir cada una de las capas que componen el perceptrón multicapa de la siguiente manera:

- **Capa de entrada:** Esta capa establece la conexión con el entorno exterior y se caracteriza por tener una neurona correspondiente a cada variable de entrada de la red. Cada neurona en esta capa recibe y procesa la información inicial.
- **Capas Ocultas:** Estas capas intermedias son agrupaciones de neuronas, donde la activación de salida de cada una resulta de una suma ponderada de las actividades de la capa anterior, además de considerar sus umbrales respectivos. Son esenciales para el procesamiento y la transformación de los datos.
- **Capa de Salida:** La capa de salida representa la última etapa de la red y se conecta con la capa oculta, generando el resultado final de la red neuronal. Produce la salida que corresponde a la predicción o clasificación que se busca obtener.

3.2.3. Ecuación estándar de una neurona

La ecuación que define el funcionamiento de una red neuronal, como se explica en la referencia [70], se compone de una combinación lineal de las variables independientes correspondientes a cada neurona, junto con sus respectivos pesos y términos de sesgo. La representación gráfica de esta ecuación de red neuronal se formula de la siguiente manera:

$$\mathbf{Z = Bias + W_1X_1 + W_2X_2 + ... + W_nX_n} \tag{1}$$

- **Z** representa el símbolo que denota la red neuronal en su conjunto.
- **W** se refiere a los pesos o coeficientes, que son parámetros ajustables de la red.
- **X** representa las variables independientes o las entradas proporcionadas a la red.

- **Bias** = W_0 donde Bias es equivalente al término de sesgo.

3.2.4. Funciones de activación

Las funciones de activación desempeñan un papel fundamental en el diseño de diversas redes neuronales. La elección de una función de activación otorga un control total sobre el proceso de entrenamiento de la red. Al incorporar estas funciones en las capas ocultas, el modelo adquiere la capacidad de aprender y realizar tareas de manera eficiente. Es crucial destacar que el tipo de predicciones que puede realizar la red neuronal depende de la función de activación seleccionada. Por lo tanto, es necesario realizar una elección cuidadosa de la función de activación que se utilizará durante el proceso de entrenamiento de la red neuronal.

3.2.4.1 Sigmoide

La primera función de activación que se aborda es la función sigmoide, la cual es una función matemática que se caracteriza por su forma en S. Algunos ejemplos comunes de funciones sigmoideas incluyen la función logística, la tangente hiperbólica y el arco tangente. Todas estas funciones comparten la propiedad de mapear un rango completo de valores reales a un intervalo más reducido, típicamente entre 0 y 1 o entre -1 y 1. Esto hace que la función sigmoide sea útil para convertir valores reales en valores que puedan interpretarse como probabilidades [71].
Las funciones sigmoideas desempeñan un papel fundamental en los modelos de regresión logística, los cuales son extensiones de la regresión lineal diseñados para la clasificación binaria. Estas funciones permiten transformar múltiples entradas reales en una probabilidad. La función sigmoide logística se define de la siguiente manera:

$$\mathbf{f(x)} = \frac{\mathbf{1}}{\mathbf{1 + e^{-x}}} = \frac{\mathbf{e^{x}}}{\mathbf{e^{x} + 1}} \tag{2}$$

La función sigmoide se utiliza como función de activación en redes neuronales, ya que toma una suma ponderada de las entradas y pasa esta suma a través de la función de activación. La salida de esta función se utiliza como entrada para la siguiente capa de la red neuronal. Dado que la función sigmoide es no lineal, la salida de la red neuronal resulta ser una función no lineal de la suma ponderada de las entradas [72].

3.2.4.2 Tangente Hiperbólica (Tanh)

La función de Tangente Hiperbólica, por otro lado, guarda similitudes con la función sigmoide, aunque con la diferencia de que los valores de salida están en el rango de -1 a 1. A pesar de compartir el problema de saturación con la función sigmoide, la función **Tanh** presenta una salida más simétrica, lo que puede facilitar el proceso de entrenamiento en redes neuronales [73].
En resumen, la función Tangente Hiperbólica produce resultados para cada valor de x proporcionado de acuerdo con la siguiente expresión:

$$\mathbf{f(x) = tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{2}{1 + e^{-2x}} - 1} \tag{3}$$

3.2.4.3 Exponencial Normalizada (SoftMax)

La función Softmax es una función matemática que transforma un vector de números en un vector de probabilidades, donde estos últimos son proporcionales a los valores relativos del vector. Esta función se utiliza cuando una red neuronal está configurada para generar **N valores**, correspondientes a **N clases** en un problema de clasificación. La función SoftMax se aplica a la salida de una capa de la red, convirtiendo los valores en probabilidades que suman uno [74], lo que restringe los valores a un rango entre 0 y 1.
La función SoftMax se define de la siguiente manera:

$$\mathbf{f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k} e^{z_k}}} \tag{4}$$

- **f(z)** representa el valor de la función SoftMax para la clase j.
- $\mathbf{e_j^z}$ es la función exponencial del valor z_j
- $\mathbf{\sum_{k=1}^{k} e^{z_k}}$ denota la suma de las funciones exponenciales de todos los valores de z_k para k desde 1 hasta N.

3.2.4.4 Unidad Lineal Rectificada (ReLU)

Otra función de activación ampliamente utilizada en el campo del aprendizaje profundo es la **Rectified Linear Unit (ReLU)**. En esta función, los valores de entrada negativos se transforman multiplicándolos por un coeficiente rectificador (generalmente 0) y dejando pasar los valores positivos sin modificarlos [75]. La función ReLU ha ganado popularidad entre los profesionales que trabajan con redes neuronales debido a sus tres ventajas principales en comparación con otras funciones de activación: simplicidad en los cálculos, propagación eficiente de la información y comportamiento lineal [76].
La simplicidad en los cálculos de la función ReLU se debe a que utiliza una operación de comparación simple con cero (a través de la función max()), en contraste con las funciones sigmoideas y la tangente hiperbólica, que involucran cálculos exponenciales más costosos. Además, la función ReLU permite una propagación eficiente de la información al generar valores cero para las entradas negativas, lo que se traduce en representaciones dispersas y simplifica tanto el modelo como el proceso de aprendizaje.
Por último, la función ReLU exhibe un comportamiento lineal que se asemeja a una función de activación lineal, optimizando el rendimiento y el comportamiento de las redes neuronales. La función ReLU se define de la siguiente manera:

$$\mathbf{f(x)=max(0,x)} = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \tag{5}$$

Es relevante señalar que existe una variante de la función ReLU conocida como Leaky ReLU. Esta variante es similar a la función ReLU, con la diferencia de que transforma los valores negativos multiplicándolos por un coeficiente rectificador pequeño (generalmente denotado como 'a'), penalizándolos mediante dicho coeficiente.

$$\mathbf{f(x)=max(0,x)} = \begin{cases} 0, & x < 0 \\ ax, & x \geq 0 \end{cases} \tag{6}$$

3.2.5. Funciones de perdida

Una función de pérdida, como menciona Ignacio Gavilán en [77], evalúa qué tan distante está la salida de una red neuronal entrenada en comparación con los resultados deseados. Esta métrica es esencial para guiar el proceso de entrenamiento, ya que proporciona información sobre la calidad de las predicciones de la red.

Dado que algunos algoritmos de aprendizaje requieren el cálculo de derivadas de la función de pérdida, esta función debe ser continua y diferenciable. Existen varias funciones de pérdida posibles, y según Ignacio Gavilán [77], algunas de las más importantes incluyen:

1. **Error Cuadrático Medio (Mean Squared Error, MSE):** Esta función se utiliza para calcular la distancia geométrica entre las predicciones de la red y los valores objetivo. El MSE puede expresarse de dos formas: elevando al cuadrado las diferencias entre las predicciones y los valores reales (**Mean Squared Error**), o tomando el valor absoluto de estas diferencias (**Absolute Error**). Estos tipos de errores son útiles en problemas relacionados con la regresión y cuando la última capa de la red no tiene una función de activación [78].

$$\mathbf{MSE} = \frac{\mathbf{1}}{\mathbf{M}} \sum_{\mathbf{i=1}}^{\mathbf{M}} (\mathbf{real_i} - \mathbf{estimado_i})^{\mathbf{2}} \tag{7}$$

2. **Entropía Cruzada Categórica (Categorical Cross Entropy):** La entropía cruzada mide la discrepancia entre dos distribuciones de probabilidad y se utiliza comúnmente en modelos de redes neuronales cuya salida representa probabilidades. Esta función de pérdida es adecuada para problemas de clasificación con múltiples clases y se combina con una función de activación [79].

$$\mathbf{L_{CE}} = -\sum_{\mathbf{i=1}} \mathbf{T_i} \log(\mathbf{S_i}) \tag{8}$$

   Donde S_i son las probabilidades por la función SoftMax y T_i son las etiquetas reales.

3. **Entropía Cruzada Binaria (Binary Cross Entropy):** Similar a la entropía cruzada categórica, esta función se utiliza en problemas de clasificación binaria, donde solo hay dos clases posibles. En este caso, la función de activación utilizada en la capa de salida es típicamente la función sigmoide [79].

$$\mathbf{L} = -\frac{\mathbf{1}}{\mathbf{N}} \left[\sum_{\mathbf{j=1}}^{\mathbf{N}} [\mathbf{t_j} \log(\mathbf{p_j}) + (\mathbf{1} - \mathbf{t_j}) \log(\mathbf{1} - \mathbf{p_j})] \right] \tag{9}$$

   Donde **N** es el número de datos, t_j representa el valor real (0 o 1) y p_j es la probabilidad calculada por la función SoftMax en el punto j.

4. **Entropía Cruzada Categórica Dispersa (Sparse Categorical Cross Entropy):** Esta variante de la entropía cruzada categórica se emplea cuando las etiquetas son números enteros que representan clases en lugar de vectores one-hot.

Estas funciones de pérdida desempeñan un papel fundamental en el proceso de entrenamiento de las redes neuronales, ya que guían la optimización de los pesos y sesgos de la red para minimizar la discrepancia entre las predicciones y los valores reales. La elección de la función de pérdida adecuada depende del tipo de problema y de la arquitectura de la red neuronal.

3.2.6. Gradiente Descendiente

El gradiente se define como el conjunto de la suma de todas las derivadas parciales de una función, y en el contexto del aprendizaje automático, el cálculo de la pendiente de la función de costo desempeña un papel fundamental. En el aprendizaje automático, uno de los principales objetivos es encontrar los parámetros $\mathbf{W}$ que minimizan la función de costo, y este proceso se aplica a una variedad de modelos, incluyendo regresión lineal, regresión polinomial, regresión logística, redes neuronales profundas y otros [78].
La técnica utilizada para estimar los mejores coeficientes de la red neuronal y, en general, para optimizar funciones, se conoce como 'gradiente descendente'. Esta técnica, basada en la optimización numérica, busca encontrar el conjunto de parámetros que minimiza la función de costo. La ecuación del gradiente [80] se define como:

$$\nabla \mathbf{f} = \left[\frac{\delta \mathbf{f}}{\delta \mathbf{x_1}}, \frac{\delta \mathbf{f}}{\delta \mathbf{x_2}}, \dots \frac{\delta \mathbf{f}}{\delta \mathbf{x_n}} \right] \tag{10}$$

Donde $\nabla \mathbf{f}$ representa el gradiente de la función $\boldsymbol{f}$ y $\frac{\delta \mathbf{f}}{\delta \mathbf{x_n}}$ representa la derivada parcial de $\boldsymbol{f}$ con respecto a la variable $\mathbf{x_n}$.El gradiente es esencial para determinar la dirección y la magnitud en la cual se deben ajustar los parámetros de la red neuronal para minimizar la función de costo.
El gradiente descendente es una herramienta fundamental en el proceso de entrenamiento de redes neuronales y en la optimización de modelos en el aprendizaje automático, ya que permite encontrar los valores óptimos de los parámetros de manera iterativa y eficiente.

3.2.7. Optimizadores

Un optimizador, como se menciona en [81], desempeña un papel fundamental en el proceso de ajuste de los valores de los parámetros de una red neuronal con el objetivo de reducir el error asociado a su funcionamiento. Este proceso se conoce como retropropagación o "backpropagation". El optimizador es responsable de realizar actualizaciones en los valores de los parámetros de manera eficiente y efectiva, con el propósito de minimizar el error cometido por la red neuronal.

La optimización más básica realizada por un optimizador consiste en la actualización gradual de los valores de los parámetros, considerando una tasa de aprendizaje predeterminada. Es importante destacar que esta actualización se basa en la derivada de la matriz de parámetros, ya que proporciona una medida del error presente en la red. El optimizador utiliza esta información para ajustar los valores de los parámetros en la dirección que minimiza el error global de la red neuronal.
En resumen, un optimizador desempeña un papel esencial en el proceso de entrenamiento de una red neuronal al actualizar sistemáticamente los valores de los parámetros para reducir el error y mejorar el rendimiento del modelo durante el aprendizaje.

3.2.7.1 Descenso de Gradiente Estocástico (Stochastic Gradient Descent,SGD

El Descenso de Gradiente Estocástico (SGD, por sus siglas en inglés) es una técnica de optimización fundamental utilizada en redes neuronales y otros algoritmos de aprendizaje automático, como se menciona en [81]. Esta estrategia se caracteriza por seleccionar aleatoriamente un punto de datos del conjunto de datos completo en cada iteración del proceso de entrenamiento. Esta elección aleatoria tiene como objetivo reducir significativamente la cantidad de cálculos necesarios para actualizar los parámetros de la red neuronal.
En el contexto del SGD, es común realizar un muestreo de una pequeña cantidad de puntos de datos en cada paso, lo que se conoce como "mini-batch gradient descent.º descenso de gradiente por lotes pequeños. Esta variante busca alcanzar un equilibrio en las ventajas del descenso del gradiente convencional y la velocidad de convergencia que ofrece el SGD [82].
La actualización de los parámetros mediante el SGD se rige por la siguiente ecuación [81]:

$$\mathbf{W} = W - \alpha * \Delta W \tag{11}$$

Donde:

- $\boldsymbol{W}$ representa la matriz de pesos.
- $\boldsymbol{\Delta W}$ es la derivada de la matriz de pesos.
- α denota la tasa de aprendizaje, que controla la magnitud de la actualización de los pesos en cada iteración.

En resumen, el Descenso de Gradiente Estocástico es una técnica de optimización que agiliza el proceso de entrenamiento de redes neuronales al seleccionar de manera aleatoria un subconjunto de datos en cada paso, lo que permite una convergencia eficiente hacia una solución óptima.

3.2.7.2 Adam

El algoritmo Adam es ampliamente reconocido en el aprendizaje profundo debido a su capacidad para combinar las ventajas de otros algoritmos de optimización, tales como el Descenso de Gradiente Estocástico, el Descenso de Gradiente con Momento (SGDM) y el Descenso de Gradiente con Momento Adaptativo (AdaGrad). Su popularidad se debe a su eficacia y competitividad en comparación con otros métodos [83].
Este algoritmo aprovecha las ventajas de los enfoques de descenso rápido y momentos adaptativos, lo que lo hace especialmente adecuado para procesos estocásticos. En el campo de la inteligencia artificial, el proceso cognitivo de la resolución de problemas exige no solo una base de datos de alta calidad para el entrenamiento, sino también una evaluación que compare las decisiones tomadas por la inteligencia natural con las de la artificial. Este proceso busca constantemente alternativas superiores [83].
La fórmula utilizada para actualizar los pesos en Adam combina elementos del Descenso de Gradiente Estocástico con Momento y del AdaGrad. Esta fórmula se expresa de la siguiente manera (ecuación 12):

$$\theta_{\mathbf{t+1}} = \theta_t - \frac{\alpha}{\sqrt{v_t} + \epsilon} m_t \tag{12}$$

Donde:

- $\theta_{\mathbf{t}}$ representa el vector de parámetros en el instante t.
- α denota la tasa de aprendizaje.
- $\mathbf{m_t}$ es la estimación del primer momento del vector gradiente.
- $\mathbf{v_t}$ es la estimación del segundo momento del vector gradiente.

Esta fórmula de actualización de pesos en Adam permite optimizar el proceso de aprendizaje y se ha convertido en una herramienta valiosa en el ámbito del aprendizaje profundo.

3.2.8. Retropropagación del Error (Backpropagation)

El proceso conocido como "backpropagation.º entrenamiento hacia atrás ocurre exclusivamente durante el entrenamiento de las redes neuronales. Consiste en el paso mediante el cual la red neuronal mejora su capacidad de hacer predicciones y se desarrolla en sentido inverso a la propagación hacia adelante. En este proceso, se calcula el error en la etapa final y se propaga desde las capas finales hacia las capas iniciales, siguiendo una dirección de derecha a izquierda [84].

3.3. Tipos de Capas

Las redes neuronales que incorporan capas recurrentes se destacan por su capacidad para trabajar con datos secuenciales o de series temporales. Estas redes encuentran aplicaciones predominantes en problemas relacionados con secuencias, como la traducción de idiomas, el procesamiento del lenguaje natural, el reconocimiento de voz e imágenes, y en sistemas de inteligencia artificial ampliamente conocidos, como Siri o el asistente de Google [85].
En su publicación [86], el autor **Martin Isaksson** identifica cuatro tipos de capas que se emplean en las redes neuronales: **Completamente Conectada** (Fully Connected), **Convolución** (Convolution), **De convolución** (Deconvolution) y **Recurrente** (Recurrent). Cada una de estas capas realiza transformaciones específicas en sus entradas, y su elección depende de la tarea en cuestión. En el contexto de capas ocultas, cada una de ellas se especializa en aprender características distintas de los datos, contribuyendo así a minimizar la función de costo global [87].

3.3.1. Densa

La capa completamente conectada, denominada así porque cada neurona de una capa se conecta con todas las neuronas de la siguiente capa, implica una carga computacional significativa. Esta característica se debe al aumento en la cantidad de conexiones a medida que se incrementa la entrada, lo que podría resultar en combinaciones potencialmente ineficientes. En este tipo de capas, cada neurona realiza transformaciones lineales sobre el vector de entrada mediante una matriz de pesos, seguidas de una transformación no lineal aplicada a través de una función de activación [88].

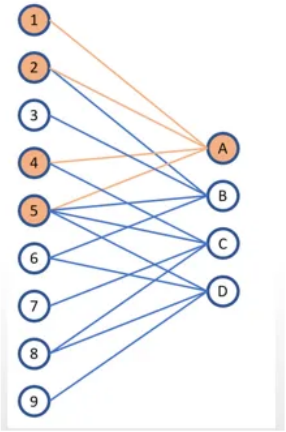


Figura 4: Representación de la Capa Densa [88]

3.3.2. Normalización por Lotes (Batch Normalization)

La técnica de normalización por lotes, conocida como “Batch Normalization”, es una estrategia que acelera la convergencia de modelos de aprendizaje profundo al reducir las desviaciones de covarianza en el conjunto de datos. Su funcionamiento implica normalizar las activaciones de entrada, transformándolas en un nuevo conjunto con una media de 0 y una desviación estándar de 1. Finalmente, se resta la media y se divide entre la desviación estándar del lote para llevar a cabo la normalización [87].

3.3.3. Agrupación (Pooling)

La combinación de capas, también conocidas como capas de ”pooling”, es una técnica fundamental para reducir la alta dimensionalidad en las redes neuronales convolucionales y tiene la capacidad de disminuir la cantidad de muestreo en los mapas de características al resumir la presencia de características en dichos mapas [87]. Estas capas se aplican generalmente después de las capas de convolución, específicamente después de que se ha aplicado una capa no lineal a un objeto, lo que resulta en una capa convolucional. La incorporación de una capa de agrupación después de una capa convolucional es uno de los patrones más comunes en la organización de capas en una red neuronal convolucional, y este patrón puede repetirse una o más veces en un modelo dado [89].

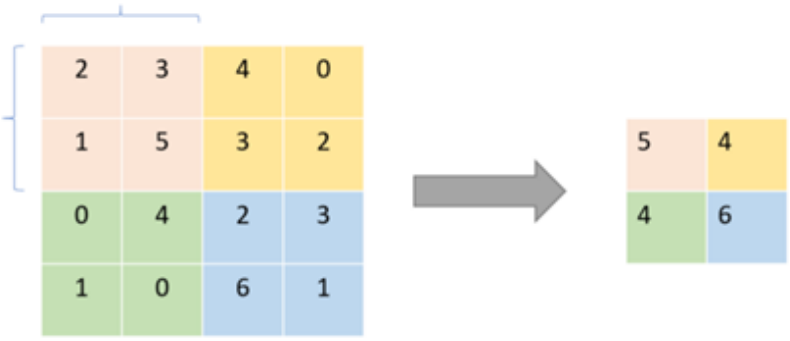


Figura 5: Representación de una capa Pooling

3.3.4. Desconexión (Dropout)

Las redes neuronales enfrentan desafíos cuando tienen una gran cantidad de entradas y un conjunto de datos pequeño, ya que esto puede llevar al modelo a aprender el ruido inherente a los datos de entrenamiento. Como resultado, el rendimiento de estas redes tiende a ser deficiente cuando se evalúan con nuevos datos. Para abordar este problema, surge la capa de dropout como técnica de regularización que tiene como objetivo mejorar el proceso de entrenamiento de las redes y reducir el sobreajuste. La capa de dropout se implementa como una capa de entrada en una red neuronal y se puede utilizar en conjunto con la mayoría de los tipos de capas. Es importante destacar que se puede aplicar en diversas capas ocultas o visibles, pero no se utiliza en las capas de salida [90]. En resumen, la capa de dropout omite de manera aleatoria neuronas en la red neuronal con el propósito de mejorar su rendimiento y generalización.

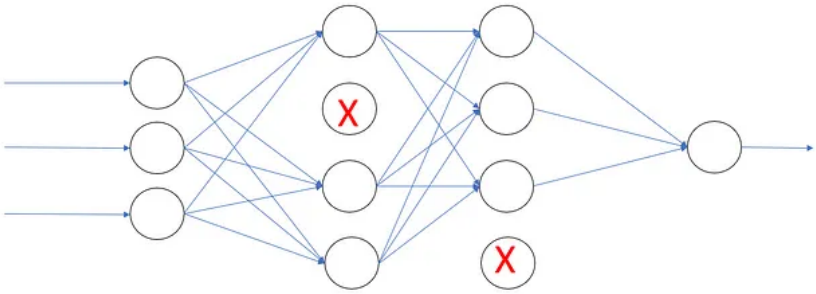


Figura 6: Ejemplos de una capa Dropout [87]

3.3.5. Capa Convolucional

Una capa de convolución es ampliamente utilizada para detectar características en imágenes, ya que utiliza un filtro para explorar pequeños conjuntos de píxeles a la vez, generando así un mapa de características que se utiliza para la clasificación [86]. La convolución implica un desplazamiento del kernel o filtro a lo largo de una matriz de entrada, calculando el producto escalar en cada posición [86], [88]. Es importante destacar que el kernel es un conjunto de pesos de n dimensiones que se multiplican con la entrada, lo que permite describir las probabilidades de que un patrón de píxeles genere una característica relevante.

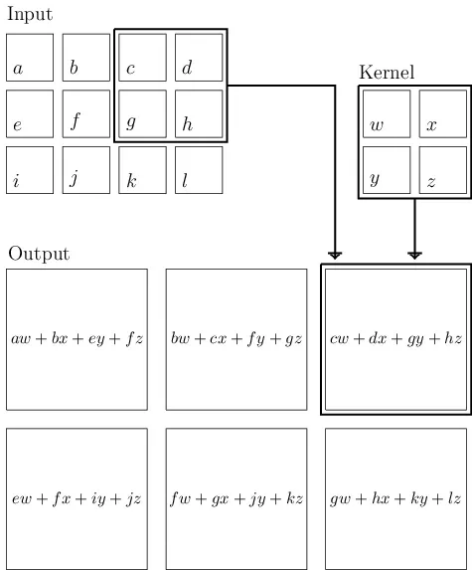


Figura 7: Representación matemática de una capa convolucional [87].

El siguiente tipo de capa es la de deconvolución, que se puede definir como una operación matemática que revierte lo realizado por una convolución. En otras palabras, esta capa es la operación inversa de una red convolucional [91]. El principal objetivo de estas capas es recuperar características o señales específicas que pueden haberse perdido previamente debido a la complejidad de otras señales o porque no se consideraban importantes para las tareas realizadas por la capa convolucional [92].

3.3.6. Flatten

Una capa de aplanamiento o **Flatten** se utiliza cuando se trabaja con entradas multidimensionales, como conjuntos de imágenes. Esta capa tiene la función de transformar los tensores de entrada multidimensionales en una sola dimensión, lo que simplifica la estructura de entrada y permite la construcción efectiva del modelo de red neuronal [93].Además, esta capa se utiliza para alisar la salida de las capas de convolución, creando así un vector de características largo que puede conectarse al modelo de clasificación, lo que a menudo se denomina capa completamente conectada o **"fully connected"** [94]. En resumen, esta capa se encarga de convertir todos los datos de las capas anteriores en una sola línea para que puedan conectarse a la última capa y ser clasificados.

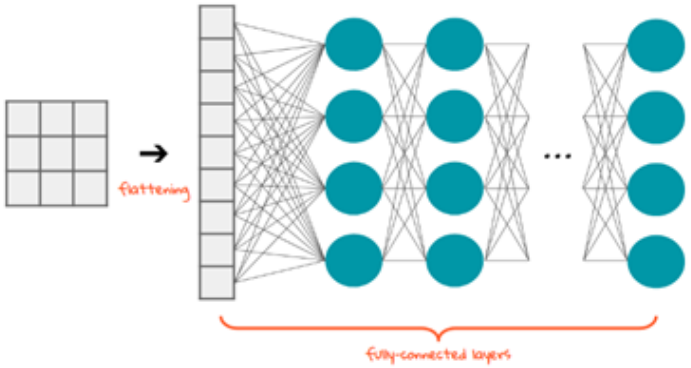


Figura 8: Representación gráfica de como una capa flatten se utiliza antes de que la información ingrese a una red neuronal [94].

3.3.7. SoftMax

La capa SoftMax es una función que transforma vectores de entrada que pueden tomar valores reales en vectores de salida que suman 1. Esta capa tiene la capacidad de procesar valores de entrada en un rango amplio, que incluye valores positivos, negativos, iguales a cero o mayores a uno. Su función principal es normalizar estos valores y mapearlos al rango de 0 a 1, de modo que puedan interpretarse como probabilidades [95]. En otras palabras, la capa SoftMax garantiza que la salida esté en forma de probabilidades, lo que facilita su interpretación.
Es importante destacar que la capa de SoftMax se utiliza exclusivamente en situaciones donde un clasificador tiene clases mutuamente excluyentes. Normalmente, se implementa como la última capa de una red neuronal, justo antes de la capa de salida, y debe tener el mismo número de nodos que esta última capa [96]. De esta manera, la capa SoftMax garantiza que la suma de las probabilidades de todas las clases sea igual a 1, lo que facilita la toma de decisiones en problemas de clasificación.

3.4. Redes Profundas

Las redes neuronales profundas se han convertido en arquitecturas fundamentales en el campo del aprendizaje profundo o **Deep Learning**. En los últimos años, estas redes artificiales han experimentado un crecimiento exponencial, ya que se han convertido en la piedra angular del reconocimiento de patrones en diversos dominios. Este avance ha impulsado significativamente el desarrollo de la inteligencia artificial, dado que las redes neuronales representan una aproximación matemática y técnica a la replicación digital de los procesos cerebrales humanos.

3.4.1. Definición

Una Red Neuronal Profunda (**Dense Neural Network, DNN**) es una variante de las Redes Neuronales Artificiales (**ANN**) que se caracteriza por contener múltiples capas ocultas ubicadas entre las capas de entrada y salida. Al igual que las **ANN** poco profundas, las **DNN** tienen la capacidad de modelar relaciones no lineales complejas en los datos.
En el contexto del aprendizaje profundo, es común que estas redes tengan una gran cantidad de capas ocultas, muchas de las cuales son no lineales. En algunos casos, el número de capas puede superar las 1000 en una única red neuronal. Para optimizar y minimizar la función de pérdida en este tipo de redes, se utiliza principalmente el método de Descenso de Gradiente.

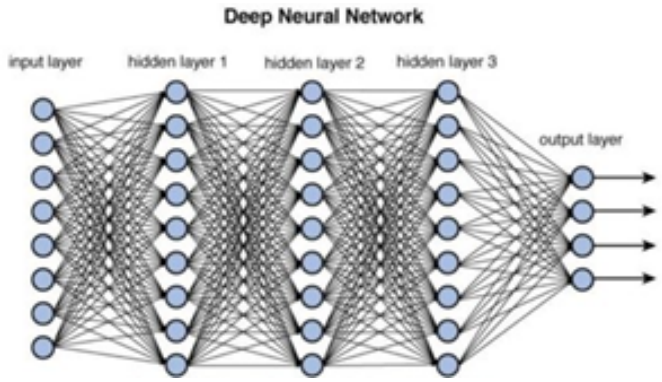


Figura 9: Visualización de una Red Neuronal Profunda [97].

3.4.2. Arquitecturas Básicas

3.4.2.1 LeNet

La arquitectura LeNet, conocida como LeNet-5, fue presentada en el trabajo de investigación titulado ***Gradient-Based Learning Applied To Document Recognition*** en el año 1998 por un equipo de científicos liderado por **Yann LeCun**, que incluía a Leon Bottou, Yoshua Bengio y Patrick Haffner. Esta red convolucional (**CNN**) se compone de un total de 7 capas, distribuidas en 3 capas convolucionales, 2 capas de submuestreo y 2 capas completamente conectadas.

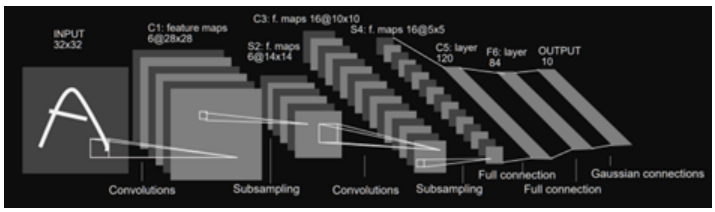


Figura 10: Visualización de la Arquitectura LeNet [98].

La primera capa de la arquitectura **LeNet-5** actúa como la capa de entrada y está diseñada para procesar imágenes de **32x32** píxeles. En el contexto del trabajo de investigación de Yann LeCun, que utilizó imágenes en escala de grises, los valores de los píxeles fueron normalizados para variar en un rango entre **-0.1 y 1.175**. Esta normalización se lleva a cabo para garantizar una media de 0 y una desviación estándar de 1 en los valores de entrada, lo que contribuye al eficiente proceso de entrenamiento.
En resumen, la arquitectura LeNet-5 se basa en dos tipos principales de capas:

- Capas Convolucionales.
- Capas de Submuestreo.

3.4.2.2 AlexNet

La arquitectura AlexNet se compone de un total de ocho capas, de las cuales cinco son capas convolucionales y tres son capas completamente conectadas. Entre las características principales utilizadas por esta red convolucional se destacan las siguientes:

1. **ReLU no Lineal**: AlexNet adopta funciones de activación ReLU (**Rectified Linear Unit**) en lugar de la función tanh convencional. Esta elección se traduce en un tiempo de entrenamiento más rápido. De hecho, las CNN que emplean ReLU pueden alcanzar un error del 25 % en el conjunto de datos en aproximadamente un sexto del tiempo necesario para las CNN que utilizan tanh.

2. **Múltiples GPUs**: AlexNet ofrece la capacidad de entrenar en múltiples GPUs al distribuir la mitad de las neuronas del modelo en una GPU y la otra mitad en CPU. Esto permite el entrenamiento de modelos más grandes y, al mismo tiempo, reduce significativamente el tiempo necesario para entrenar las redes neuronales.
3. **Overlapping Pooling**: A diferencia de las CNN tradicionales que fusionan las salidas de grupos adyacentes de neuronas que no se superponen, AlexNet introduce la técnica de pooling con superposición (**overlapping pooling**). Aunque esto inicialmente hacía que el ajuste de los modelos fuera más desafiante, se observó una reducción del error aproximadamente un 0.5 % al adoptar esta técnica.

Estas características son fundamentales en la arquitectura de AlexNet y han contribuido significativamente a su éxito en aplicaciones de visión por computadora.

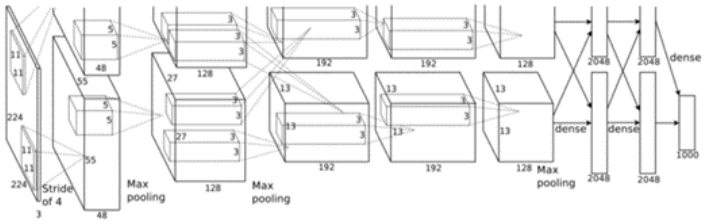


Figura 11: Visualización de la arquitectura AlexNet [99].

3.4.2.3 VGGNet

La arquitectura VGG, desarrollada por **K. Simonyan** y **A. Zisserman** de la Universidad de Oxford en 2014, es una destacada Convolutional Neural Network (CNN) que surgió como respuesta al desafío de reconocimiento visual a gran escala de **ImageNet (ILSVRC)**. VGG representa dos variantes principales: **VGG16 y VGG19**, que se distinguen principalmente por la cantidad de capas que las componen, siendo 16 y 19, respectivamente.
En la arquitectura **VGG16**, como se detalla en [100], la entrada debe ser una imagen RGB con un tamaño fijo de 224x224 para su procesamiento inicial en la primera capa convolucional. A partir de ahí, la imagen atraviesa una serie de capas convolucionales que utilizan filtros de 3x3. Además, en una de estas capas se emplean filtros de 1x1, que funcionan como una transformación lineal de los canales de entrada. La arquitectura también incluye tres capas totalmente conectadas, con las dos primeras conteniendo 4096 canales cada una y la tercera con 1000 canales, seguidas de una capa softmax para la clasificación final. Es importante destacar que todas las capas ocultas utilizan la función de activación ReLU.

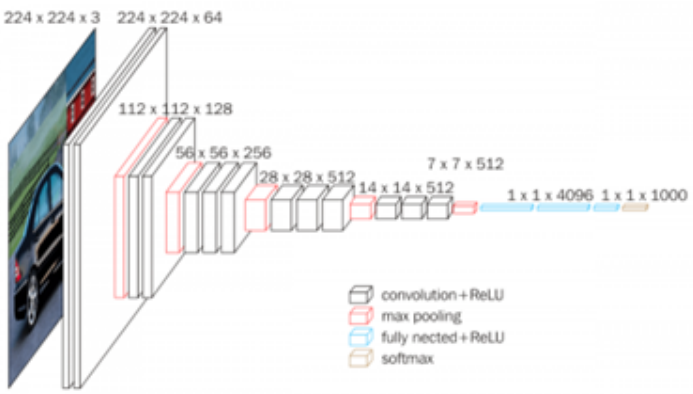


Figura 12: Visualización de la arquitectura VGGNet [100].

La arquitectura VGG se ha destacado en el ámbito del procesamiento de imágenes y la clasificación de objetos debido a su profundidad y capacidad para extraer características de las imágenes de manera efectiva.

3.4.2.4 ResNet

La Red Residual, comúnmente conocida como ResNet, es otra arquitectura ampliamente utilizada en el campo del aprendizaje profundo. Esta red se ha desarrollado para abordar el problema del gradiente que desaparece mediante la introducción de un enfoque innovador denominado "**bloques residuales**", junto con una técnica conocida como "**omisión de conexiones**". Esta técnica establece conexiones directas entre las activaciones de una capa y las de capas posteriores, lo que resulta en la creación de bloques residuales. La característica central de ResNet es permitir que la red se ajuste a un mapeo residual en lugar de obligar a cada capa a aprender el mapeo subyacente. De manera más formal, esto se puede expresar como $\mathbf{F}(\mathbf{x}) := \mathbf{H}(\mathbf{x}) - \mathbf{x}$,lo que significa que $\mathbf{H(x)}$ se define como la salida de un bloque residual que representa la diferencia entre la salida esperada ($\mathbf{H(x)}$) y la entrada ($\mathbf{x}$).

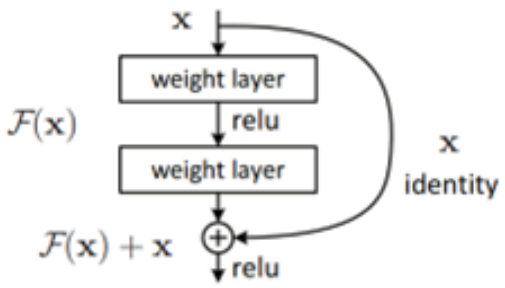


Figura 13: Representación visual de un bloque residual [100].

3.4.3. Redes Residuales

La Red Neuronal Residual, también conocida como **ResNet**, es una arquitectura de red artificial reconocida que logra realizar sus tareas de clasificación mediante atajos o conexiones de salto que cruzan varias capas. En el campo de las redes neuronales profundas, los expertos suelen implementar redes residuales con conexiones de salto que se extienden a través de dos o tres capas, y estas conexiones incluyen normalización por lotes y operaciones no lineales entre ellas. En algunos casos, se emplean matrices de peso adicionales para aprender los pesos asociados a los saltos.
La razón principal detrás de la omisión de capas en estas redes es evitar el problema del desvanecimiento del gradiente y los desafíos relacionados con él. Dado que el gradiente se retropropaga hacia las capas anteriores durante el entrenamiento, el método de salto de capas ayuda a mantener el gradiente lo suficientemente grande al retroceder en la red. No obstante, este enfoque de saltar capas funciona de manera eficaz principalmente cuando las capas intermedias son completamente lineales o se superponen con una capa no lineal.

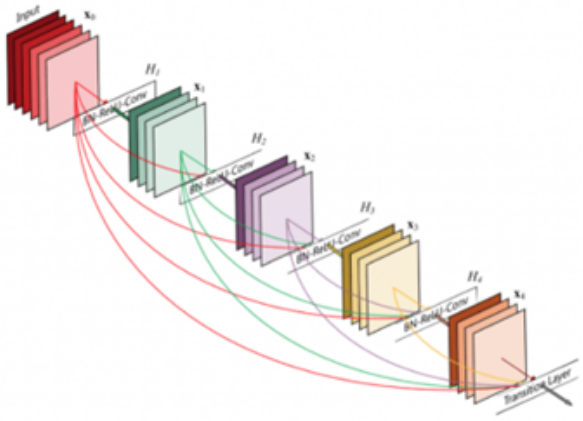


Figura 14: Arquitectura de una Red Residual [101].

3.5. Modelo y Evaluación

Al abordar las métricas de evaluación en el contexto del aprendizaje automático, es importante destacar que estas métricas están estrechamente relacionadas con las tareas de clasificación y regresión. Estas dos tareas son ejemplos de aprendizaje supervisado, y la elección de métricas adecuadas desempeña un papel crucial en la evaluación del rendimiento de los modelos. La utilización de métricas apropiadas contribuye a mejorar la capacidad predictiva de nuestro modelo, ya que permite una evaluación más completa y precisa. Sin embargo, si nos limitamos únicamente a medir la precisión, existe el riesgo de obtener predicciones incorrectas [102].

3.5.1. Verdadero Positivo, Verdadero Negativo, Falso Positivo,Falso Negativo

Una matriz de confusión, también conocida como matriz de error, es una herramienta fundamental en la evaluación del rendimiento de un modelo de clasificación. Esta tabla permite comparar las predicciones realizadas por el modelo con los valores reales conocidos, y aunque suele tener una estructura básica de 2x2, puede expandirse para abordar distintas categorías. Dentro de una matriz de confusión binaria, se encuentran los siguientes clasificadores:

- **Falso Positivo (FP)**: Representa una predicción positiva por parte del modelo cuando el valor verdadero es negativo.
- **Falso Negativo (FN)**: Indica una predicción negativa del modelo cuando el valor verdadero es positivo.
- **Verdadero Positivo (VP)**: Corresponde a una predicción positiva del modelo que coincide con un valor verdadero positivo.
- **Verdadero Negativo (VN)**: Se refiere a una predicción negativa del modelo que coincide con un valor verdadero negativo.

3.5.2. Precisión y Error

La precisión, también conocida como **Accuracy**, es una métrica utilizada para evaluar la frecuencia con la que un clasificador realiza predicciones correctas. Se define como la relación entre el número de predicciones correctas y el número total de predicciones realizadas. Para que la precisión sea una métrica útil, es importante que las clases de destino estén equilibradas, es decir, que el número de datos de entrenamiento sea aproximadamente igual para cada clase correspondiente, lo que garantiza que el modelo no esté sesgado.
La fórmula de precisión se expresa como:

$$\textbf{Precision} = \frac{VP + VN}{VP + VN + FP + FN} \tag{13}$$

3.5.3. Métricas de Sensibilidad-Especificidad

La sensibilidad, también conocida como "Sensitivity.º Recall", mide cuántos de los casos positivos reales son capaces de predecir correctamente nuestro modelo. Esta métrica resulta especialmente útil en situaciones donde los falsos negativos son más críticos que los falsos positivos.
La fórmula de sensibilidad se expresa como:

$$\textbf{Sensibilidad} = \frac{VerdaderoPositivo}{VerdaderoPositivo + FalsoNegativo} \tag{14}$$

3.5.4. Métricas de Recuperación de Precisión

La precisión es una métrica que evalúa la proporción de casos clasificados correctamente como positivos en relación con todos los casos clasificados como positivos. Esta métrica es particularmente relevante en situaciones donde los falsos positivos tienen un impacto más significativo que los falsos negativos.
La precisión se utiliza comúnmente en sistemas de recomendación, como recomendación de artículos, música, sitios web, entre otros, donde errores en la recomendación pueden resultar en la pérdida de clientes y ventas. Se calcula dividiendo los verdaderos positivos entre la suma de los verdaderos positivos y falsos positivos:

$$\textbf{Precisión} = \frac{Verdadero\ Positivo}{Verdadero\ Positivo + Falso\ Positivo} \tag{15}$$

Además, el valor de F se obtiene a partir de una combinación de las métricas de precisión y sensibilidad. El valor máximo de F se alcanza cuando la precisión es igual a la sensibilidad y se calcula de la siguiente manera:

$$\textbf{F} = 2 * \frac{Precisión * Sensibilidad}{Precisión + Sensibilidad} \tag{16}$$

3.5.5. Hiperparámetros

La configuración adecuada de hiperparámetros desempeña un papel de vital importancia en el éxito de entrenamiento de modelos en el aprendizaje automático. Estos valores, en su mayoría, no se derivan de los datos, sino que suelen ser especificados por el científico de datos o el ingeniero de machine learning. El valor óptimo de un hiperparámetro no puede conocerse de antemano para un problema específico. En su lugar, se recurre a valores genéricos que han funcionado en problemas similares o se busca la mejor opción mediante un proceso de prueba y error [103].
Entre los hiperparámetros fundamentales se encuentran el número de pasos por época y los pasos de validación, que funcionan como el timón que guía el proceso de entrenamiento y evaluación de las redes. La elección incorrecta de estos valores puede dar lugar a problemas como el sobreajuste, donde el modelo se adapta en exceso a los datos de entrenamiento, o el sobreentrenamiento, donde no logra capturar patrones importantes. Por lo tanto, lograr la configuración adecuada de estos hiperparámetros representa un desafío crítico que enfrentan los profesionales del aprendizaje profundo.

3.5.5.1 Pasos por Época

Este valor se emplea durante el proceso de entrenamiento y se refiere a la frecuencia con la que se procesa un lote de datos en una época de entrenamiento. El cálculo de este valor depende tanto del tamaño del conjunto de datos de entrenamiento como del tamaño del lote (batch size) que se esté utilizando. Puede calcularse utilizando la siguiente fórmula:

$$\textbf{Número de pasos por época} = \frac{Tamaño\ del\ conjunto\ de\ datos\ de\ entrenamiento}{Tamaño\ de\ lote} \tag{17}$$

Donde:

- **Tamaño del conjunto de datos de entrenamiento:** representa el número total de ejemplos en el conjunto de datos utilizado para entrenar el modelo.
- **Tamaño de lote:** indica la cantidad de ejemplos que se procesan en cada paso durante el entrenamiento.

3.5.5.2 Pasos de Validación

Este valor se emplea durante la validación del modelo, generalmente después de cada época de entrenamiento, con el propósito de evaluar su desempeño. Al igual que con el número de pasos por época, el cálculo depende del tamaño del conjunto de datos de validación y del tamaño de lote que se esté utilizando. Puede calcularse mediante la siguiente fórmula:

$$\textbf{Número de pasos por validación} = \frac{Tamaño\ del\ conjunto\ de\ datos\ de\ validación}{Tamaño\ de\ lote} \tag{18}$$

Donde:

- **Tamaño del conjunto de datos de validación:** representa el número total de ejemplos en el conjunto de datos utilizado para validar el modelo.
- **Tamaño de lote:** indica la cantidad de ejemplos que se procesan juntos en cada paso durante la validación.

3.6. Librerías y Frameworks

Existen diversas bibliotecas que permiten desarrollar redes neuronales en el lenguaje de programación Python, y entre las más populares se encuentran **NumPy, Pandas, Scikit-Learn y TensorFlow**. A continuación, se describirá la importancia de cada una de estas bibliotecas y cómo se utilizan en el ámbito de las redes neuronales.

3.6.1. NumPy

NumPy es un proyecto de código abierto que tiene como objetivo habilitar la computación numérica en Python, proporcionando un tipo de dato que representa matrices multidimensionales y las operaciones necesarias para manipularlas [104]. Esta biblioteca se especializa en el cálculo numérico y el análisis masivo de datos, lo que la convierte en una herramienta esencial para tareas que implican manipulación y procesamiento de datos numéricos.
Una ventaja significativa de utilizar NumPy en comparación con las listas predefinidas en Python radica en su eficiencia computacional. El procesamiento de arreglos con NumPy es hasta 50 veces más rápido que el procesamiento de listas, lo que lo convierte en una elección óptima para trabajar con vectores y matrices de dimensiones considerables [105].

La principal estructura de datos en la biblioteca NumPy son los arreglos (arrays). Estos arreglos son conjuntos de datos del mismo tipo organizados en una tabla o cuadrícula de diferentes dimensiones. El nombre que se utiliza para describir la cantidad de dimensiones de un arreglo varía según el contexto. Por ejemplo, un arreglo de una sola dimensión se denomina vector, mientras que uno con dos dimensiones se llama matriz. En casos de tres dimensiones, se hace referencia a ellos como cubos, y así sucesivamente. NumPy proporciona las herramientas necesarias para crear, manipular y operar con estos arreglos de manera eficiente en Python.

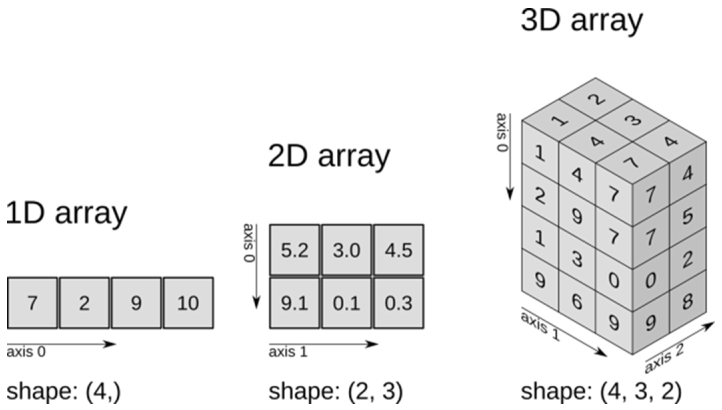


Figura 15: Representación de un arreglo una sola dimensión (izquierda), dos dimensiones (centro) y tres dimensiones (derecha) [124].

3.6.2. Pandas

Pandas es un paquete de Python que proporciona estructuras de datos rápidas, flexibles y expresivas diseñadas para el manejo de datos tabulares. Esta biblioteca es ampliamente reconocida por su capacidad para manipular y analizar datos de manera más flexible y poderosa que otros lenguajes, gracias a su versatilidad para trabajar con diversos tipos de datos [106].
Entre las funciones clave de Pandas se encuentran:

- El manejo de datos tabulares con columnas de datos heterogéneos, similares a una tabla SQL o una hoja de cálculo de Excel.
- La capacidad para trabajar con datos de series de tiempo, tanto ordenados como desordenados.
- La manipulación de datos en forma de matrices arbitrarias con etiquetas de fila y columna.
- La capacidad para manejar cualquier conjunto de datos estadísticos, sin necesidad de etiquetarlos previamente.

Pandas se basa en dos tipos de datos principales: **Series** (una sola dimensión) y **DataFrames** (dos dimensiones). Estos tipos de datos son ampliamente utilizados en campos como finanzas, estadísticas, ciencias sociales y diversas disciplinas de ingeniería. Pandas está diseñado para integrarse con NumPy, lo que la convierte en una herramienta esencial en el entorno de la informática científica [107].
En resumen, Pandas es una herramienta versátil que simplifica la carga, modelado, análisis y manipulación de datos de diversas fuentes. Facilita la preparación de datos para su posterior uso y análisis, lo que la convierte en una biblioteca fundamental para tareas de procesamiento y análisis de datos.

3.6.3. Scikit-Learn

La biblioteca scikit-learn, también conocida como sklearn, es una destacada librería de código abierto y una de las más populares en Python para tareas de aprendizaje automático. Sklearn ofrece una amplia gama de herramientas eficientes para el aprendizaje automático y el modelado estadístico, abordando tareas que incluyen clasificación, regresión, agrupación y reducción de dimensiones [108]. Una de las ventajas clave de scikit-learn es su estrecha integración con otras bibliotecas de Python ampliamente utilizadas, como **NumPy, SciPy y Matplotlib**. Esta integración permite el desarrollo de modelos de manera eficaz y eficiente en el contexto del aprendizaje automático y la ciencia de datos [109]. La biblioteca ofrece una variedad de módulos y algoritmos que resultan fundamentales en diferentes etapas del proceso de análisis de datos y modelado. Entre los modelos populares proporcionados por scikit-learn se incluyen:

- **Algoritmos de Aprendizaje Supervisado:** Regresión lineal, máquinas de vectores de soporte, árboles de decisión, entre otros.
- **Algoritmos de Aprendizaje No Supervisados:** Análisis Factorial, PCA (Análisis de Componentes Principales), Redes Neuronales no supervisadas.
- **Agrupación:** Utilizada para agrupar datos sin etiquetas en clústeres o grupos similares.
- **Reducción de Dimensionalidad:** Reduce la cantidad de atributos en los datos, lo que facilita la síntesis, visualización y selección de características [110].

3.6.4. TensorFlow y Keras

TensorFlow, desarrollado por el equipo de **Google Brain** y lanzado en 2015, es una biblioteca de código abierto ampliamente utilizada en el ámbito del aprendizaje automático y el desarrollo de redes neuronales en Python [111]. Esta librería desempeña un papel crucial al realizar cálculos numéricos que respaldan tareas de aprendizaje automático y proporciona herramientas para la adquisición y procesamiento de datos en modelos de entrenamiento, permitiendo la realización de predicciones y la mejora de los procesos para futuros resultados.
TensorFlow incluye una amplia variedad de modelos y algoritmos para el aprendizaje automático y profundo, utilizando principalmente Python o JavaScript como lenguajes de programación. La biblioteca ofrece una API conveniente que facilita la creación de aplicaciones que abarcan desde la clasificación de dígitos escritos a mano hasta el reconocimiento de imágenes, palabras o secuencias de palabras. Además, TensorFlow cuenta con una extensa colección de modelos ya entrenados que pueden ser utilizados en diversos proyectos.
Keras, por su parte, es una API de aprendizaje profundo en Python que se utiliza principalmente en TensorFlow para tareas de aprendizaje automático [112]. Esta API fue desarrollada con el objetivo de proporcionar una herramienta eficiente para experimentar en el campo del aprendizaje profundo, permitiendo a los investigadores pasar rápidamente de la idea al resultado.
Keras se destaca por ser simple, flexible y potente:

- **Simple:** Keras reduce la carga cognitiva para los programadores, permitiéndoles centrarse en las partes esenciales del problema.
- **Flexible:** Keras adopta el principio de revelación progresiva de la complejidad, lo que significa que los flujos de trabajo deben ser sencillos y claros, con la posibilidad de abordar desafíos más complejos a medida que se avanza en el proceso.
- **Potente:** Keras ofrece un rendimiento y escalabilidad notables, siendo utilizada por organizaciones de renombre como NASA, YouTube y Waymo.

Keras se relaciona estrechamente con TensorFlow 2, ya que es una API de alto nivel que proporciona una interfaz accesible y productiva para la resolución de problemas de aprendizaje automático. Además, **Keras** permite a los desarrolladores aprovechar la escalabilidad y la capacidad de **TensorFlow 2** en diversas plataformas, incluidas unidades de procesamiento tensorial (TPUs) y grandes conjuntos de unidades de procesamiento gráfico (GPUs), y exportar soluciones a entornos como navegadores web y dispositivos móviles.

3.6.5. Puntos Clave

La detección de puntos clave (keypoints) implica la ubicación precisa de elementos fundamentales en los objetos analizados. Por ejemplo, en el contexto facial, los puntos clave pueden abarcar desde las puntas de la nariz y las cejas hasta las comisuras de los ojos, entre otros. Estos puntos clave desempeñan un papel crucial al enriquecer la representación del objeto subyacente. Las aplicaciones de la detección de puntos clave son diversas e incluyen la estimación de la postura corporal y la detección de rostros [113].

Figura 16: Ejemplos utilizados en [113] para mostrar cómo se ven los puntos clave.

3.6.6. Anotaciones de Imágenes

La detección de objetos se refiere a la habilidad de un sistema computacional para identificar y localizar objetos de un tipo específico dentro de una imagen o escena dada [114]. Para llevar a cabo esta tarea, se requiere que los datos de entrenamiento se representen en archivos XML o archivos JSON. Cada uno de estos formatos posee sus propias ventajas y desventajas.

3.6.6.1 COCO

El conjunto de datos **Microsoft Common Objects in Context (COCO)** [115] es ampliamente reconocido y empleado en el campo de la detección de objetos, siendo una referencia común para la evaluación del rendimiento de diversos métodos de visión artificial. Este conjunto de datos utiliza un formato especial en estructura **JSON** para definir la representación de las características y los metadatos de un conjunto de imágenes.
Algunas de las características destacadas del conjunto de datos COCO incluyen [116], [117]:

- Anotaciones detalladas de instancias para la segmentación de objetos.
- Reconocimiento en contexto de objetos.
- Segmentación de elementos de superpíxeles.
- Más de 200,000 imágenes etiquetadas de un total de 330,000 imágenes.
- Un total de 1.5 millones de instancias de objetos.
- 80 categorías de objetos, incluyendo instancias individuales que pueden ser fácilmente etiquetadas.
- 91 categorías de elementos, que incluyen materiales y objetos que no poseen límites claramente definidos.
- Información contextual significativa proporcionada.
- Cinco subtítulos disponibles por cada imagen.
- Un conjunto de 250,000 anotaciones de personas, que incluye 17 puntos clave diferentes ampliamente utilizados para estimar poses.

Estas características hacen que el conjunto de datos COCO sea una valiosa herramienta para la investigación y el desarrollo en el campo de la visión por computadora y la detección de objetos.

3.6.6.2 PASCAL VOC

PASCAL, que significa **Análisis de Patrones, Modelado Estadístico y Aprendizaje Computacional** (Pattern Analysis, Statistical Modelling, and Computational Learning, PASCAL) [114], fue el escenario de varios desafíos de detección de objetos entre los años 2005 y 2012, utilizando una estructura de archivos estandarizada para almacenar las anotaciones de las imágenes.
El desafío PASCAL Visual Object Classes (VOC) comprendía dos componentes principales [114]:

- Un conjunto de datos público junto con un software de evaluación estandarizado.
- Un concurso anual y un taller dedicados a la detección de objetos.

Los objetivos principales de este desafío eran evaluar la capacidad de los modelos en las siguientes áreas [114]:

- **Clasificación:** determinar si un objeto estaba presente en una imagen.
- **Detección:** localizar la posición de los objetos dentro de la imagen.

3.6.6.3 LabelImg

LabelImg es una herramienta de anotación de imágenes que permite a los usuarios definir visualmente cuadros delimitadores alrededor de objetos en imágenes y, posteriormente, guarda automáticamente estas anotaciones en archivos XML. Esta aplicación proporciona una solución sencilla y de libre acceso para llevar a cabo el etiquetado de fotografías [118]. Es relevante señalar que LabelImg se utiliza en el entorno del lenguaje de programación Python 3.

4. DESARROLLO Y METODOLOGÍA

En este capítulo, se detalla la metodología que se sigue para llevar a cabo el proyecto de reconocimiento de señas estáticas en la Lengua de Señas Mexicana (LSM), en línea con los objetivos establecidos. Las metodologías propuestas se organizan en fases secuenciales para garantizar un desarrollo riguroso y eficaz del proyecto.
La primera etapa consiste en una preparación técnica exhaustiva relacionada con las redes neuronales. Esto implica adquirir un conocimiento profundo sobre las diversas variantes de redes neuronales, comprender su implementación en diferentes lenguajes de programación y analizar qué tipos de datos son compatibles con ellas. El objetivo principal de esta fase es seleccionar un modelo de red neuronal que se adecue a los objetivos específicos del proyecto.
La siguiente metodología se centra en la investigación y selección de un conjunto de datos apropiado para el proyecto, específicamente uno que contenga imágenes relacionadas con la LSM. Se lleva a cabo un análisis detallado para identificar las señas estáticas y dinámicas, determinar el tamaño de las imágenes y examinar otras características clave.
Una vez definidos tanto el modelo de red neuronal como el conjunto de datos correspondiente, se procede a buscar y aplicar una arquitectura que se ajuste a los requisitos previamente establecidos. Esto incluye decisiones sobre el número de capas intermedias, capas de salida, capas convolucionales y otros aspectos relacionados.
La siguiente fase está dedicada a la validación e implementación del modelo, la arquitectura y la base de datos en conjunto. Se llevan a cabo pruebas exhaustivas para identificar posibles problemas de compatibilidad u otros desafíos que puedan surgir durante la implementación. En caso de encontrar problemas, se realizan investigaciones adicionales para localizar y resolver los errores presentes.
Finalmente, se enfoca en el desarrollo de una aplicación que integre la arquitectura de red neuronal, la base de datos y el modelo previamente definidos. Esta aplicación se diseña para dispositivos que ejecutan sistemas operativos Windows u otro entorno compatible con el lenguaje de programación Python. La aplicación tiene la capacidad de reconocer señas estáticas a través de la cámara del dispositivo o desde una base de datos proporcionada.
En conclusión, estas metodologías se aplican secuencialmente para llevar a cabo el proyecto de reconocimiento de señas estáticas en la LSM, garantizando un desarrollo riguroso y una implementación exitosa. Cada fase se lleva a cabo con meticulosidad para garantizar la calidad y eficacia del proyecto en su conjunto.

4.1. Preparación de los Datos

Para iniciar la recolección de datos, es esencial definir qué información es relevante recopilar en este proyecto. Los datos fundamentales por recolectar se refieren a imágenes relacionadas con la Lengua de Señas Mexicana (LSM). El Consejo Nacional para Prevenir la Discriminación (CONAPRED) proporciona un recurso que contiene las señas correspondientes a todo el abecedario de la LSM.

En consecuencia, las letras que se excluyen de la búsqueda son aquellas relacionadas con las letras **J, K, Ñ, Q, X y Z**. Las letras no mencionadas en esta lista son las que se consideran necesarias para el proyecto. Sin embargo, es importante señalar que, a pesar de que existen bases de datos relacionadas con el lenguaje de señas, como la **Lengua de Señas Americana (ASL)**, **Lengua de Señas Española (SSL)**, **Lengua de Señas Francesa (FSL)**, entre otras, no se dispone de una base de datos pública relacionada específicamente con la LSM. Por lo tanto, se requerirá la creación de una base de datos propia que contenga imágenes que representen las letras de la LSM, con un enfoque particular en aquellas en las que la palma de la mano sea visible y en diferentes posiciones.
Para obtener los conjuntos de datos necesarios, será necesario buscar en diversas fuentes aquellas que cumplan con los requisitos establecidos. Dado que la base de datos más comúnmente disponible en línea está relacionada con la ASL, se utilizarán conjuntos de datos de este lenguaje de señas como recurso principal. El primer paso en la preparación de datos consistirá en identificar las similitudes que existen entre las diferentes bases de datos y la LSM. A continuación, se presenta una tabla que muestra las señas similares entre la LSM y la ASL, FSL y SSL, donde el símbolo "**X**" representa que se encontró una coincidencia con la LSM y el símbolo "**-**" significa totalmente lo contrario.

MSL(LSM)	ASL	FSL	SSL	MSL	ASL	FSL	SSL
A	–	X	–	**N**	–	–	–
B	X	X	–	**Ñ**	–	–	–
C	X	X	X	**O**	X	X	–
D	X	X	–	**P**	–	–	–
E	X	X	X	**Q**	–	–	–
F	X	X	–	**R**	X	X	X
G	–	–	–	**S**	X	X	–
H	–	–	–	**T**	X	–	–
I	X	X	X	**U**	X	X	–
J	–	X	–	**V**	X	X	–
K	–	–	–	**W**	X	X	–
L	X	X	X	**X**	–	–	–
M	–	–	–	**Y**	–	–	–
				Z	X	X	–

Cuadro 1: Similitudes entre la LSM con ASL, FSL y SSL

Este proceso de identificación de señas similares permite seleccionar las imágenes adecuadas de las bases de datos de cada lengua de señas que puedan utilizarse en el proyecto de reconocimiento de señas estáticas en LSM.

4.2. Primer Conjunto de Datos

Dado que no hay un conjunto de datos público disponible para la Lengua de Señas Mexicana (LSM), es necesario construir un conjunto de datos propio. Utilizando la información presentada en la Tabla 1, que compara las diferentes lenguas de señas, procedemos a crear el primer conjunto de datos a partir de conjuntos de datos públicos disponibles en Kaggle, proporcionados por Kapil Londhe [119], y el conjunto de datos público de Ayushi Thakur, disponible en [120]. En total, recopilamos 77,073 imágenes representando las letras B, C, E, F, I, L, O, R, S, T, U, V y W.

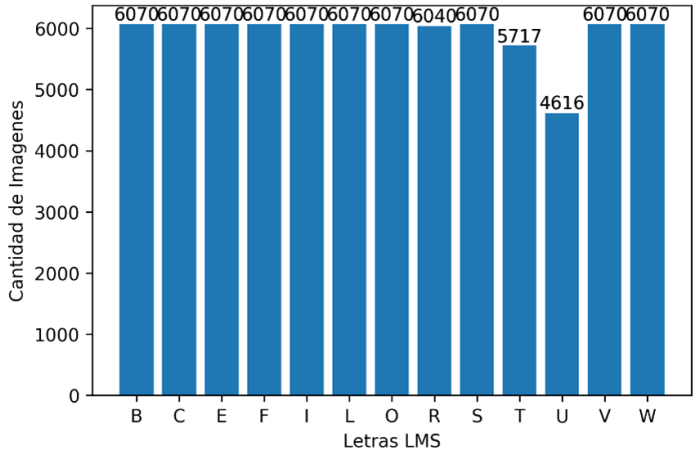


Figura 17: Desglose del número de imágenes por categoría del primer conjunto de datos.

Es importante señalar que existe un claro desequilibrio entre las categorías en términos de la cantidad de imágenes recolectadas. Para llevar a cabo las primeras pruebas de manera efectiva, es necesario balancear el conjunto de datos para que cada categoría cuente con un número de muestras manejable.

4.3. Conjunto de datos para primeras pruebas

Debido al desequilibrio observado en el primer conjunto de datos, se tomó la decisión de limitar el número de imágenes por categoría a un total de 100 imágenes por cada una. Este ajuste se realiza con el propósito de llevar a cabo pruebas iniciales utilizando redes neuronales y evaluar la idoneidad del conjunto de datos para su uso en el proyecto. Posteriormente, se procede a revisar todas las imágenes para asegurarse de que cumplan con los siguientes requisitos:

1. Visibilidad de manos y dedos en las imágenes.
2. Posición adecuada de las manos en las imágenes.
3. Tamaño uniforme de las imágenes, con dimensiones de 400x400 píxeles y 3 canales de color (RGB).

Estas características son de vital importancia, ya que la claridad de las imágenes, la visibilidad de los dedos y la correcta posición de las manos son factores determinantes para obtener resultados concluyentes en las pruebas. Asimismo, el tamaño uniforme de las imágenes garantiza la consistencia de los datos y facilita su procesamiento.

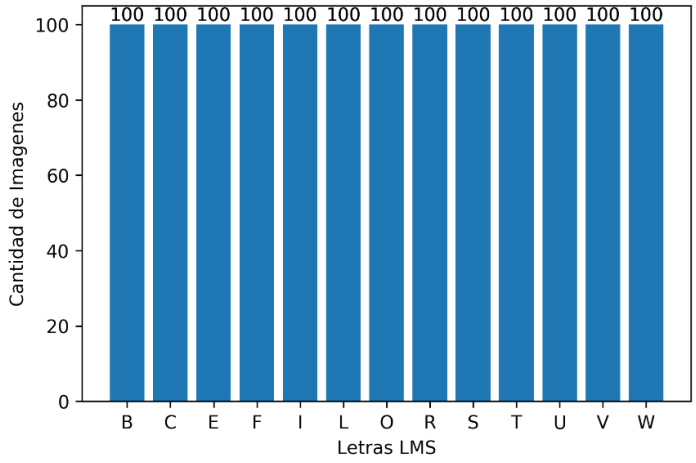


Figura 18: Desglose del número de imágenes por categoría para las primeras pruebas.

4.3.1. Modelo para las primeras pruebas

En las primeras pruebas, se desarrolla un modelo de aprendizaje utilizando el lenguaje de programación Python y se emplean redes neuronales convolucionales (CNN) para su entrenamiento. Este modelo se construye haciendo uso de dos bibliotecas principales: Matplotlib y Keras.Matplotlib es una biblioteca que proporciona herramientas para la creación de visualizaciones estáticas, animadas e interactivas en Python. Permite generar gráficos de alta calidad y ofrece una forma eficiente y sencilla de visualizar datos. Por otro lado, Keras es una biblioteca específicamente diseñada para la implementación de redes neuronales en Python y es compatible con el framework TensorFlow. De esta biblioteca, se importan varios componentes esenciales para el entrenamiento del modelo y el procesamiento de las imágenes obtenidas.

En primer lugar, se importa la clase **Sequential**, que actúa como una estructura secuencial que permite la conexión de capas de la red neuronal de forma secuencial, facilitando así la construcción del modelo. A continuación, se importa la función **Conv2D**, que crea un kernel de convolución para generar un tensor de salida. También se utilizan otras funciones relacionadas con las capas, como **Activation**, que se utiliza para especificar la función de activación de las neuronas, y **Dense**, que establece una capa densamente conectada donde cada neurona está conectada a todas las neuronas de la capa siguiente.
Es importante destacar que, para las primeras pruebas, se emplean tres capas convolucionales con la función de activación **ReLU** y una capa **Flatten** para aplanar los datos. Además, se implementa una función que aplica transformaciones a las imágenes, como redimensionamiento y cambio de orientación, con el objetivo de mejorar el proceso de entrenamiento del modelo.

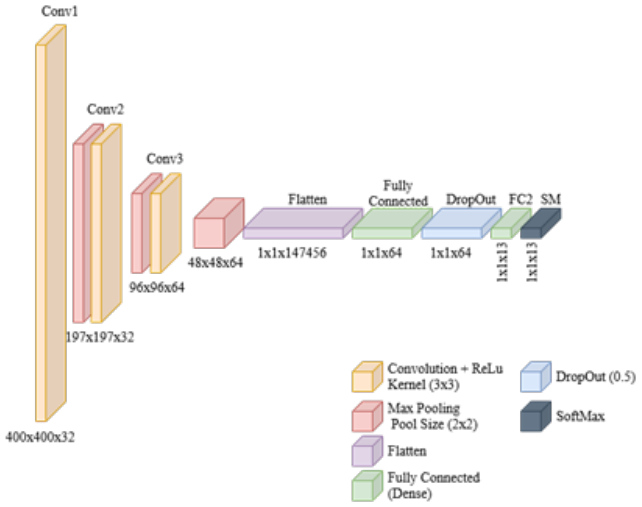
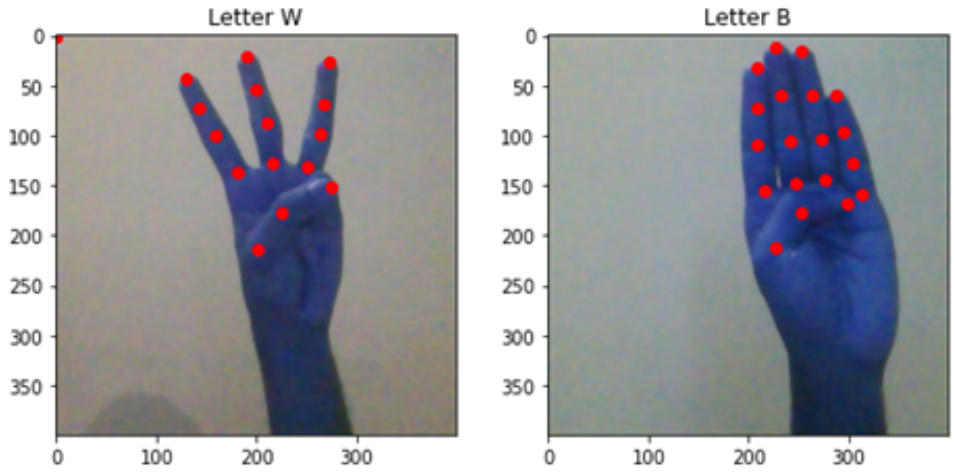


Figura 19: Arquitectura implementada para las primeras pruebas.

4.4. Conjunto de datos para pruebas finales

Después de realizar las primeras pruebas y ajustar la cantidad de imágenes, se decide limitar tanto el número de categorías como de imágenes para poder aplicar la técnica de **keypoints** o puntos clave en el reconocimiento de señas. Sin embargo, surge un desafío adicional, dado que, aunque se cuenta con una base de datos considerable, no se dispone de una base de datos que incluya imágenes con puntos clave relevantes para el reconocimiento de señas, lo que plantea un nuevo obstáculo.

Para abordar este problema, se recurre a la herramienta **LabelImg** para generar imágenes junto con sus puntos clave y así cumplir con los objetivos del proyecto. Sin embargo, debido a la falta de experiencia previa en la creación y utilización de puntos clave en imágenes para redes neuronales, se opta por limitar el número de categorías a cinco: las letras B, C, E, L y W. En consecuencia, cada categoría contiene un total de 8**0 imágenes para entrenamiento** y **20 para validación**, sumando un total de **400 imágenes para entrenamiento** y **100 para validación**.
Es importante destacar que se manejan un total de **19 pares** de puntos clave, los cuales consisten en coordenadas X y Y para representar los dedos de la mano. Estos puntos se dividen en tres para el dedo pulgar y cuatro para el resto de los dedos, representando así las articulaciones del dedo y la punta del mismo. Esta decisión se toma con el propósito de mejorar el reconocimiento de las señas, ya que algunas señas muestran claramente los 19 puntos, mientras que en otras faltan puntos debido a que los dedos están doblados hacia la palma o no son visibles en la señal.



(a) Dedo Oculto (b) Dedos Visibles

Figura 20: Ejemplo de una seña con un dedo oculto (**a**) y una seña con todos los dedos visible (**b**).

4.4.1. Modelos para las pruebas finales

Para estas pruebas finales, se emplearon dos arquitecturas de redes neuronales ampliamente reconocidas y probadas en el campo de visión por computadora: VGG16 y ResNet50. Estas arquitecturas son conocidas por su eficacia en la extracción de características de imágenes y han demostrado un rendimiento sólido en una variedad de tareas de reconocimiento visual. Su elección se basa en la necesidad de obtener representaciones profundas y precisas de las señas estáticas en la lengua de señas mexicana (LSM).

Para asegurar que el modelo de redes neuronales tenga acceso a los puntos clave necesarios para la detección y el reconocimiento de señas, se crearon archivos JSON específicos para cada categoría de señas y para las imágenes de entrenamiento y validación respectivamente. Cada archivo JSON contiene las coordenadas exactas de los puntos clave correspondientes a las señas representadas en las imágenes.
Esta metodología de almacenamiento y organización de datos permite mantener un enfoque estructurado y coherente en las últimas pruebas. Además, garantiza que el modelo pueda acceder de manera eficiente a los puntos clave necesarios para su entrenamiento y validación, ya que las coordenadas están vinculadas directamente a cada imagen correspondiente. El proceso de adquisición y organización de los puntos clave mediante archivos JSON es fundamental para garantizar la consistencia y la precisión en el reconocimiento de las señas de LSM.
Además de modificar la arquitectura principal de la red neuronal a **VGG16** y **ResNet50**, se realizaron cambios significativos en las salidas de la red para abordar de manera efectiva la tarea de reconocimiento de señas. En lugar de tener una única salida para la clasificación de las señas, se optó por una estructura de salida dual, como se visualiza en la **Figura 20**. Es importante destacar que, para el caso de ResNet50, solo es necesario sustituir el cuadro correspondiente a la arquitectura VGG16.

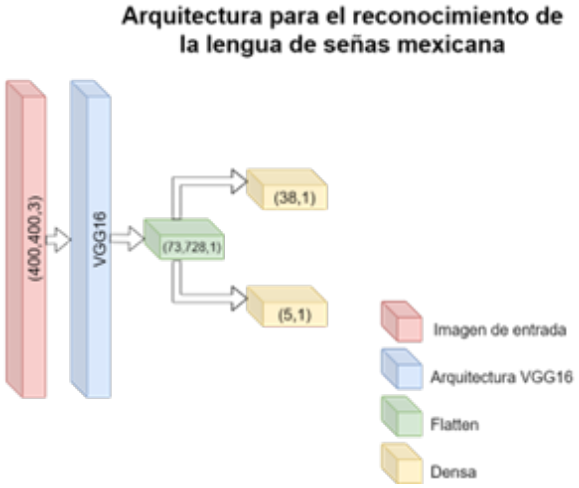


Figura 21: Arquitectura para VGG16 y ResNet50.

Por un lado, la red neuronal está diseñada para generar cinco salidas distintas, cada una correspondiente a una de las cinco categorías de señas: B, C, E, L y W. Estas salidas permiten la clasificación precisa de las señas estáticas, identificando a qué clase pertenecen y proporcionando una base sólida para el reconocimiento inicial de las señas. Por otro lado, se incorporaron treinta y ocho salidas adicionales, correspondientes a las coordenadas en los ejes X e Y de los diecinueve puntos clave utilizados para representar los dedos de la mano. Estas salidas permiten que la red neuronal detecte y localice con precisión la posición de los puntos clave en la imagen, lo que es esencial para comprender y reconocer las señas.

Esta estructura de salida dual proporciona una solución integral para la tarea de reconocimiento de señas en LSM. Los resultados y el rendimiento de la red neuronal con esta configuración se detallarán en la sección de resultados y discusión, lo que permitirá una comprensión más completa de la efectividad de este enfoque en la tarea propuesta. En resumen, a lo largo de este proyecto, se llevaron a cabo una serie de pasos fundamentales, desde la recopilación de datos hasta la implementación de redes neuronales avanzadas y cambios significativos en la estructura de salida, con el objetivo de desarrollar un sistema de reconocimiento de señas en LSM preciso y eficiente.

5. RESULTADOS Y DISCUSIÓN

En esta sección, se presentan los resultados obtenidos y se lleva a cabo una discusión sobre el desempeño y la eficacia del sistema de reconocimiento de señas en Lengua de Señas Mexicana (LSM) desarrollado en este proyecto. Se analizan los resultados de las pruebas realizadas utilizando las arquitecturas de redes neuronales VGG16 y ResNet50, así como la estructura de salida dual diseñada para la detección de señas estáticas y la localización de puntos clave en las imágenes. Se evalúa la precisión y la eficiencia del sistema en la clasificación de las señas, así como su capacidad para detectar y localizar correctamente los puntos clave. Además, se discuten las implicaciones de estos resultados y se ofrecen posibles áreas de mejora y futuras direcciones de investigación en el campo del reconocimiento de señas en LSM.

5.1. Resultados de las primeras pruebas

Para las pruebas iniciales, se llevan a cabo un total de seis experimentos con el primer modelo propuesto. En las primeras cinco pruebas, se comparte un conjunto común de parámetros, incluido el tamaño de lote (**batch size**) establecido en 10 unidades tanto para el entrenamiento como para la validación. Además, se fija un total de **50 épocas** con **5 pasos por época**. Es relevante señalar que, en estas primeras cinco pruebas, se realizan variaciones en los parámetros, como se detallan en el **Cuadro 2**.En cuanto a las pruebas número 6, se realizaron ajustes en varios parámetros, incluyendo la cantidad de épocas y los pasos por época, obteniendo los resultados de la **Cuadro 3**.

Prueba	**Rescale**	**Zoom**	**Width Shift**	**Height Shift**	**Shear Range**	**Horizontal Flip**	**Fill Mode**	**Validation Split**
Prueba 1	1/255	–	–	–	–	–	–	0.2
Prueba 2	1/255	0.15	0.2	–	–	–	–	0.2
Prueba 3	1/255	–	–	0.2	0.15	–	–	0.2
Prueba 4	1/255	–	–	–	–	True	Nearest	0.2
Prueba 5	1/255	0.15	0.2	0.2	0.15	True	Nearest	0.2

Cuadro 2: Primeras 5 pruebas con el primer modelo.

Las pruebas con diferentes configuraciones de parámetros, como las mostradas en el **Cuadro 2**, se realizaron para evaluar el impacto de diversas transformaciones de imagen en el rendimiento del modelo de reconocimiento de señas. Estas transformaciones, como el escalado, el desplazamiento, el sesgo, el volteo horizontal, entre otros, pueden ayudar a aumentar la diversidad del conjunto de datos y mejorar la capacidad del modelo para generalizar a nuevos ejemplos. Al ajustar estos parámetros, podemos explorar cómo diferentes transformaciones afectan la capacidad de aprendizaje del modelo y su capacidad para reconocer señas estáticas en la Lengua de Señas Mexicana (LSM).

Prueba	**Rescale**	**Validation Split**	**Horizontal flip**	**Fill Mode**	**Steps per Epoch**	**Validation Steps**
Prueba 6	1/255	0.2	True	Nearest	10	10

Cuadro 3: Prueba 6 con el primer modelo

La tabla presenta los resultados de la prueba número 6, en la cual se realizaron ajustes específicos en varios parámetros. El parámetro **Rescale** indica que las imágenes se están escalando a una fracción de su tamaño original, lo que puede ayudar a mejorar la eficiencia del entrenamiento del modelo. La validación se realiza con un valor de división del 20 %, lo que garantiza una evaluación adecuada del rendimiento del modelo. La opción de **Horizontal Fli**p está activada, lo que permite que las imágenes se roten horizontalmente durante el entrenamiento, aumentando la diversidad del conjunto de datos y mejorando la generalización del modelo. El **Fill Mode** se establece en **Nearest**, lo que indica que se utilizará el valor del píxel más cercano para completar cualquier espacio vacío creado durante las transformaciones de la imagen. Además, se especifica que habrá **10 pasos por época** durante el entrenamiento, lo que puede proporcionar una cobertura más completa del conjunto de datos y una mejor convergencia del modelo. El número de **pasos de validación** también se establece en **10** para garantizar una evaluación precisa del rendimiento del modelo en cada época. En conjunto, estos ajustes buscan mejorar la capacidad del modelo para reconocer y generalizar las señas estáticas en la lengua de señas mexicana.

Después de realizar una serie de experimentos con diferentes configuraciones de parámetros, se han obtenido resultados significativos en términos de precisión y pérdida en el proceso de entrenamiento de modelos de reconocimiento de señas estáticas en la lengua de señas mexicana (LSM). Los resultados se presentan en la **Cuadro 4**, donde se muestra la exactitud (**Accuracy**) y la pérdida (**Loss**) para cada una de las pruebas realizadas.

Prueba	Accuracy	Loss
Prueba 1	87 %	0.44
Prueba 2	74 %	0.98
Prueba 3	41 %	1.73
Prueba 4	84 %	0.68
Prueba 5	19 %	2.08
Prueba 6	82 %	0.76

Cuadro 4: Valores de la exactitud (Accuracy) y de la pérdida (Loss) de las diferentes pruebas.

Dado los resultados del **Cuadro 4**, se puede concluye lo siguiente:

- **Prueba 1**: Se alcanzó una precisión del 87 %, lo que indica que el modelo clasificó correctamente el 87 % de las señas estáticas en LSM. La pérdida asociada fue de 0.44, lo que sugiere que el modelo tuvo un rendimiento aceptable en la minimización de errores durante el entrenamiento.
- **Prueba 2**: La precisión disminuyó significativamente a un 74 %, lo que sugiere que el modelo tuvo dificultades para generalizar patrones en los datos de validación. La pérdida aumentó a 0.98, lo que indica que el modelo tuvo un rendimiento inferior en comparación con la Prueba 1.
- **Prueba 3**: La precisión fue aún menor, alcanzando solo el 41 %, lo que sugiere que el modelo tuvo dificultades significativas para clasificar correctamente las señas estáticas. La pérdida aumentó considerablemente a 1.73, lo que indica que el modelo experimentó una alta tasa de errores durante el entrenamiento.
- **Prueba 4**: Se observó una mejora en la precisión, alcanzando el 84 %, lo que indica que el modelo pudo recuperarse parcialmente de las dificultades encontradas en las pruebas anteriores. La pérdida disminuyó a 0.68, lo que sugiere una mejora en la capacidad del modelo para ajustarse a los datos de entrenamiento.
- **Prueba 5**: La precisión disminuyó drásticamente a solo el 19 %, lo que sugiere un rendimiento deficiente del modelo en la clasificación de señas estáticas. La pérdida aumentó significativamente a 2.08, lo que indica que el modelo experimentó dificultades importantes durante el entrenamiento.
- **Prueba 6**: Se observó una precisión del 82 %, lo que sugiere una recuperación del rendimiento del modelo en comparación con la Prueba 5. La pérdida se redujo a 0.76, lo que indica una mejora en la capacidad del modelo para ajustarse a los datos de entrenamiento.

Estos resultados resaltan la importancia de ajustar cuidadosamente los parámetros del modelo y de realizar pruebas exhaustivas para mejorar su rendimiento en la tarea de reconocimiento de señas estáticas en LSM. En este contexto, considerando los resultados obtenidos, la Prueba 1 emerge como la más prometedora debido a su alta precisión del **87 %** y una pérdida relativamente baja de **0.44**. Esta prueba demostró la capacidad del modelo para clasificar correctamente las señas estáticas con un bajo nivel de error.

5.2. Resultados de las pruebas finales

En las pruebas finales, se realiza una evaluación exhaustiva de los modelos de reconocimiento de la lengua de señas presentados con anterioridad, destacando que, aunque en la etapa anterior se realizan intervenciones directas en las imágenes, en estas últimas pruebas, no se realizan dichas intervenciones. Esto significa que no se aplican transformaciones como cambios en el tamaño, rotaciones o ajustes en la orientación. Esta decisión se basa en la necesidad de evaluar el desempeño de los modelos en condiciones cercanas a las del entorno real, manteniendo la integridad de los datos de entrada sin ninguna manipulación deliberada.
Para lograr este objetivo, se emplea la ecuación que determina el número de pasos por época (**17**) y el número de pasos por validación (**18**), teniendo en cuenta el tamaño del lote (batch size) establecido en **10 unidades**. En esta configuración, los datos de entrenamiento constan de un total de **400 imágenes**, mientras que el conjunto de validación comprende **100 imágenes**. Esta estrategia de diseño nos permite llevar a cabo una evaluación precisa y equilibrada del rendimiento de nuestros modelos, específicamente utilizando un total del **80 %** de las imágenes para entrenamiento y **20 %** para la validación.
Además, en esta etapa de pruebas, se realizan evaluaciones separadas para cada una de las arquitecturas de red utilizadas, específicamente, VGG16 y ResNet50. El único parámetro variable durante las evaluaciones es la tasa de aprendizaje (learning rate). Esta variación permite comprender en detalle cómo este factor influye en el rendimiento de los modelos.

5.2.1. VGG16

En el contexto de la primera arquitectura, VGG16, se llevan a cabo un total de cinco experimentos, enfocados en tres aspectos críticos que inciden en el proceso de entrenamiento del modelo. Estos aspectos comprenden la disposición de las imágenes de entrada, si se encuentran organizadas por categoría o no, las tasas de aprendizaje empleadas y el número de épocas de entrenamiento. Los resultados obtenidos se segmentan y analizan en función de los siguientes indicadores de desempeño: Exactitud de las Coordenadas (Coordinates Accuracy), Exactitud de las Categorías (Label Accuracy), Pérdida de Coordenadas (Coordinates Loss), Pérdida de Categorías (Label Loss) y el Tiempo de Entrenamiento de la Arquitectura (Time), tal como se presenta en detalle en la **Cuadro 5**.

Test	Shuffle	Learning Rate	Epochs	Coordinates Accuracy	Label Accuracy	Coordinates Loss	Label Loss	Time (s)
1	NO	0.001	5	47 %	71 %	1495.15	1.04	–
2	NO	0.0005	5	32 %	62 %	1386.89	4.18	–
3	NO	0.0001	5	66 %	100 %	42.14	1.7E-07	4262
4	YES	0.0001	5	71 %	100 %	69.91	1.72E-06	6178
5	YES	0.0001	7	70 %	100 %	81.98	2.38E-09	9102

Cuadro 5: Pruebas realizadas con la arquitectura VGG16

En términos generales, **las pruebas 4 y 5**, que incluyen un ordenamiento aleatorio de las imágenes y una tasa de aprendizaje de 0.00001, arrojan los resultados más sobresalientes en cuanto a precisión tanto en la predicción de coordenadas como en la clasificación de categorías. Este hallazgo sugiere que la inclusión de un ordenamiento aleatorio de las imágenes durante el entrenamiento puede tener un impacto positivo en el rendimiento de los modelos.
En lo que respecta al tiempo de entrenamiento, es evidente que las pruebas que implican ordenamiento aleatorio y un mayor número de épocas (**pruebas 4 y 5**) demandan más tiempo de procesamiento. Esta consideración es esencial, especialmente en aplicaciones prácticas donde la eficiencia temporal es un factor crítico.
Sin embargo, al comparar las pruebas 4 y 5, se advierte que la prueba 4 se destaca como la opción preferible. Aunque el tiempo de entrenamiento es menor en la prueba 4 en comparación con la prueba 5, aún se mantiene un rendimiento superior en términos de precisión en las coordenadas y clasificación de categorías. Por lo tanto, se determina que la prueba 4 (**Test 4**) es la configuración más adecuada para la comparación con el siguiente modelo.
Esta selección se basa en la búsqueda de un equilibrio óptimo entre precisión y eficiencia temporal, lo que garantiza un rendimiento sólido sin incurrir en tiempos de entrenamiento excesivos. La prueba 4 servirá como punto de referencia para evaluar y comparar el próximo modelo en el proceso de mejora continua.

5.2.2. ResNet50

Como en el caso anterior con la arquitectura VGG16, se mantendrán los mismos indicadores de evaluación: Exactitud de las Coordenadas (Coordinates Accuracy), Exactitud de las Categorías (Label Accuracy), Pérdida de Coordenadas (Coordinates Loss), Pérdida de Categorías (Label Loss) y el Tiempo de Entrenamiento de la Arquitectura (Time). Esto proporcionará una comparación exhaustiva entre ambas arquitecturas y permitirá identificar cuál de ellas es más adecuada para el problema de reconocimiento de señas.

En el caso de ResNet50, se llevarán a cabo un total de siete pruebas, de las cuales tres se realizarán sin ordenamiento de las imágenes y cuatro incluirán el ordenamiento aleatorio. Además, se realizarán variaciones en la tasa de aprendizaje y en el número de épocas para explorar cómo estos parámetros afectan el rendimiento del modelo.
A través de estas pruebas, se busca identificar las configuraciones óptimas para la arquitectura ResNet50, lo que permitirá tomar decisiones informadas sobre su uso en el contexto del reconocimiento de señas. Los resultados se presentarán y analizarán detalladamente para determinar cuál de las configuraciones y modelos generales es el más prometedor y efectivo para abordar este desafío específico.

Test	**Shuffle**	**Learning Rate**	**Epochs**	**Coordinates Accuracy**	**Label Accuracy**	**Coordinates Loss**	**Label Loss**	**Time (s)**
1	NO	0.001	5	77 %	99 %	594.42	0.06	4059
2	NO	0.0005	5	80 %	100 %	194.37	0.0014	4201
3	NO	0.0001	5	80 %	100 %	1343.41	2.01E-06	4124
4	YES	0.0001	5	75 %	100 %	1166.54	1.51E-06	2309
5	YES	0.0001	7	78 %	100 %	931.75	4.89E-08	3241
6	YES	0.0005	10	75 %	100 %	154.53	0.00E+0	4117
7	YES	0.0005	20	85 %	100 %	65.16	0.00E+0	9502

Cuadro 6: Pruebas realizadas con la arquitectura ResNet50

Dada la información presentada en el **Cuadro 6**, es evidente que **las pruebas 2, 3 y 7** destacan por su notable desempeño en términos de la precisión de las coordenadas, superando el 80 % de exactitud en esta métrica. Sin embargo, al analizar la pérdida de coordenadas, se observa que las **pruebas 2 y 3** tienen valores que superan las 1000 unidades, lo que indica una mayor pérdida en la detección de coordenadas. Por otro lado, **la prueba 7** se destaca positivamente, ya que no solo tiene una pérdida menor en comparación con las pruebas 2 y 3, sino que también registra valores por debajo de las 100 unidades, lo cual es altamente favorable.
Es importante señalar que, al comparar todas las pruebas en términos de precisión, la mayoría de ellas alcanzan el 100 % en la exactitud de las categorías y se acercan significativamente a una pérdida cercana a cero. Esto indica que la arquitectura **ResNet50** es altamente efectiva para identificar a qué clase pertenece cada imagen, lo que es esencial para la detección de las letras de la Lengua de Señas Mexicana.

No obstante, al considerar el desempeño global, la **prueba 5** se destaca como la más sobresaliente entre todas las pruebas. A pesar de su excelente precisión y pérdida reducida, es importante tener en cuenta que esta configuración requiere un tiempo de entrenamiento significativamente mayor, superando los **9500 segundos**. Esta información es crucial para la implementación práctica del modelado, ya que se debe equilibrar el rendimiento con el tiempo de entrenamiento en un entorno real.

5.3. Comparación Resultados de VGG16 y ResNet50

En esta sección se procede a llevar a cabo una comparación de los resultados más destacados obtenidos a través de las dos arquitecturas de redes neuronales empleadas en este estudio: VGG16 y ResNet50. Los datos resultantes se someten a un análisis meticuloso, considerando la precisión en la detección de coordenadas y categorías, así como la función de pérdida asociada y el tiempo de entrenamiento requerido. Esta evaluación tiene como propósito determinar cuál de las dos arquitecturas presenta un rendimiento superior en la tarea fundamental de identificar las letras de la Lengua de Señas Mexicana.
Las comparaciones iniciales se centran en analizar las pruebas óptimas obtenidas para cada una de las arquitecturas, evaluando las características fundamentales previamente presentadas en las tablas (**Exactitud de Coordenadas, Exactitud de Categoría, Pérdida de Coordenadas y Pérdida de Categoría**). Además, se realiza una comparativa de la tasa de aprendizaje, el número de épocas y el tiempo de entrenamiento, ya que este último depende intrínsecamente de los dos primeros.
Es importante destacar que, a partir de la **Figura 22** hasta la **Figura 25**, se representan los resultados previamente mencionados, utilizando la gráfica naranja para representar la arquitectura VGG16 (con 5 épocas de entrenamiento) y la gráfica azul para ResNet50 (con 20 épocas de entrenamiento). La significativa diferencia en la cantidad de épocas se debe a que la arquitectura ResNet50 demuestra ser considerablemente más eficiente en términos de tiempo por época, dando una ventana más amplia de entrenamiento para esta arquitectura.
En una segunda instancia, se aborda una comparación relativa al número total de imágenes que cada modelo logra predecir con precisión en cada categoría. Esta métrica permite discernir si alguno de los modelos exhibe un rendimiento superior en este aspecto o si ambos alcanzan resultados satisfactorios.
Por último, se lleva a cabo una evaluación visual de la precisión de los modelos en la predicción de los puntos clave de las manos y la identificación de a qué dedo corresponden dichos puntos. Esta comparación visual es esencial para determinar la exactitud en la representación de las señas, lo que permite distinguir entre señas correctas e incorrectas.

5.3.1. Mejores Pruebas

Como se indicó previamente en la sección anterior, en esta etapa inicial de comparación, se examinan las pruebas óptimas obtenidas para cada arquitectura, evaluando diversas características previamente mencionadas. El **Cuadro 7** ilustra esta comparación inicial, y a continuación, se profundiza en la discusión de cada característica, destacando sus aspectos relevantes.

Arquitectura	Test	Shuffle	Learning Rate	Epochs	Coordinates Accuracy	Label Accuracy	Coordinates Loss	Label Loss	Time (s)
VGG16	4	YES	0.0001	5	71 %	100 %	69.91	1.72E-06	6178
ResNet50	7	YES	0.0005	20	85 %	100 %	65.16	0.00E+0	9502

Cuadro 7: Tabla comparativa de las mejores pruebas de cada arquitectura

A partir de los resultados presentados en la tabla anterior, se derivan las siguientes conclusiones:

1. **Mejor Desempeño en la Precisión de Coordenadas (Coordinates Accuracy)**: La arquitectura ResNet50 (prueba 7) muestra un rendimiento superior en la precisión de la predicción de coordenadas en comparación con VGG16 (prueba 4). ResNet50 logra un notable 85 % de precisión, mientras que VGG16 alcanza el 71 %. Este resultado indica que ResNet50 es más eficiente en la localización de puntos clave en las imágenes.

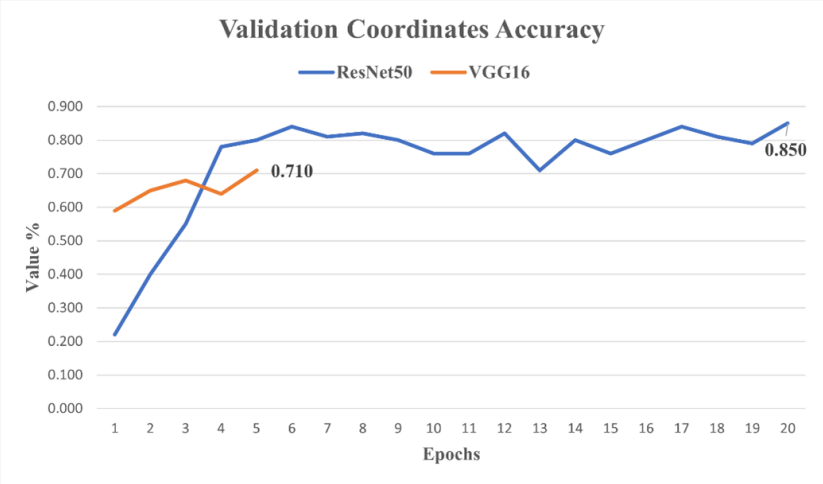


Figura 22: Comparación en la precisión de coordenadas en las arquitecturas utilizadas

2. **Excelente Precisión en la Clasificación de Categorías (Label Accuracy)**: Ambas arquitecturas logran una precisión del 100 % en la predicción de categorías (Label Accuracy). Esto confirma que ambas arquitecturas son altamente efectivas en la clasificación precisa de imágenes en sus respectivas categorías.

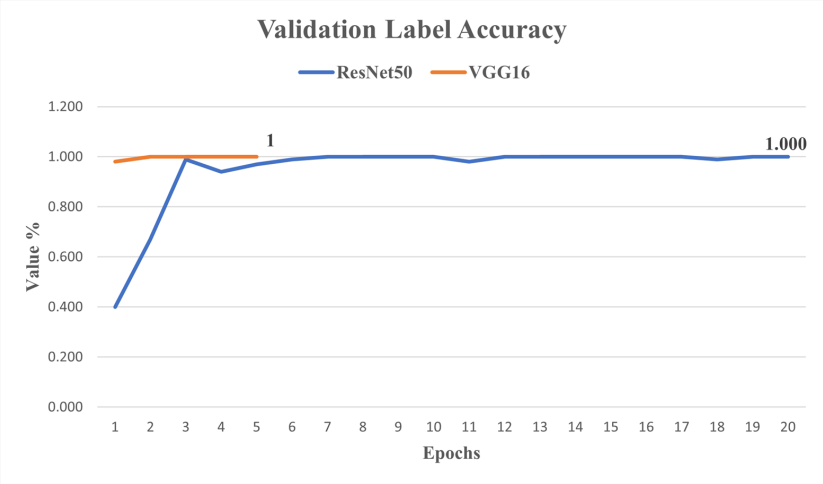


Figura 23: Comparación en la precisión de clasificación de categorías en las arquitecturas utilizadas.

3. **Menor Pérdida en Coordenadas (Coordinates Loss)**: La prueba 7 de ResNet50 exhibe una pérdida en coordenadas significativamente menor (65.16) en comparación con la prueba 4 de VGG16 (69.91). Este resultado destaca la mayor precisión de ResNet50 en la estimación de las coordenadas de los puntos clave en las imágenes.

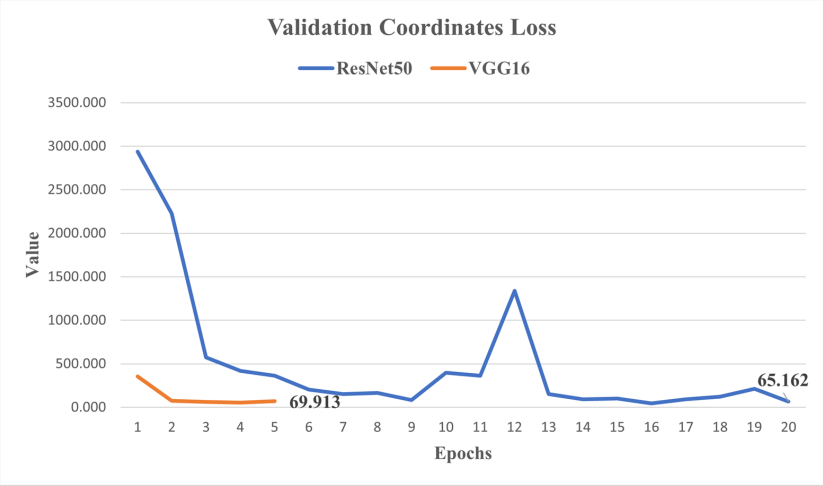


Figura 24: Comparación en la perdida en las coordenadas en las arquitecturas utilizadas.

4. **Menor Pérdida en la Clasificación de Categorías (Label Loss):** Como se puede observar en la Cuadro 7, al analizar la pérdida en la clasificación de categorías, ambas arquitecturas presentan valores muy cercanos a cero. La Figura 32 proporcionará una representación gráfica de este proceso y permitirá una visualización más detallada de cómo VGG16 alcanza una pérdida de cero en un tiempo eficiente en comparación con ResNet50.

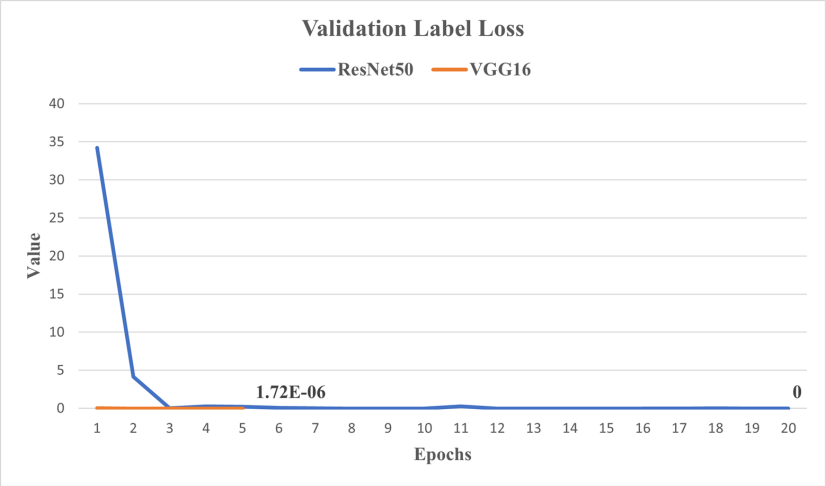


Figura 25: Comparación en la perdida en la clasificación de categorías en las arquitecturas utilizadas.

5. **Mayor Tiempo de Entrenamiento en ResNet50**: ResNet50 (prueba 7) requiere un tiempo de entrenamiento más extenso (9502 segundos) en comparación con VGG16 (prueba 4, 6178 segundos). Esto indica que ResNet50 demanda un mayor tiempo de entrenamiento, lo cual debe ser considerado en proyectos que requieran una rápida implementación.

En resumen, ResNet50 supera a VGG16 en términos de precisión en coordenadas y pérdida en coordenadas, lo que la posiciona como una opción más adecuada para tareas de detección de puntos clave. Sin embargo, es importante tener en cuenta que ResNet50 también requiere un mayor tiempo de entrenamiento. La elección entre ambas arquitecturas dependerá de las necesidades específicas del proyecto, buscando un equilibrio entre precisión y tiempo de entrenamiento.

5.3.2. Predicciones de categoría de cada arquitectura

Como se mencionó anteriormente, para llevar a cabo las pruebas utilizando las diferentes arquitecturas previamente entrenadas, se emplea un conjunto de datos compuesto por un total de 50 imágenes, con 10 imágenes correspondientes a cada una de las cinco categorías. Estas imágenes se utilizan para realizar predicciones tanto de la categoría a la que pertenecen como de los puntos clave de las señas. Es importante destacar que, en ambos casos, tanto para la arquitectura VGG16 como para ResNet50, las predicciones generadas y representadas en un mapa de calor son comparables en ambas arquitecturas. Esto sugiere que ambas arquitecturas demuestran una excelente capacidad para predecir con precisión las categorías de las imágenes evaluadas.

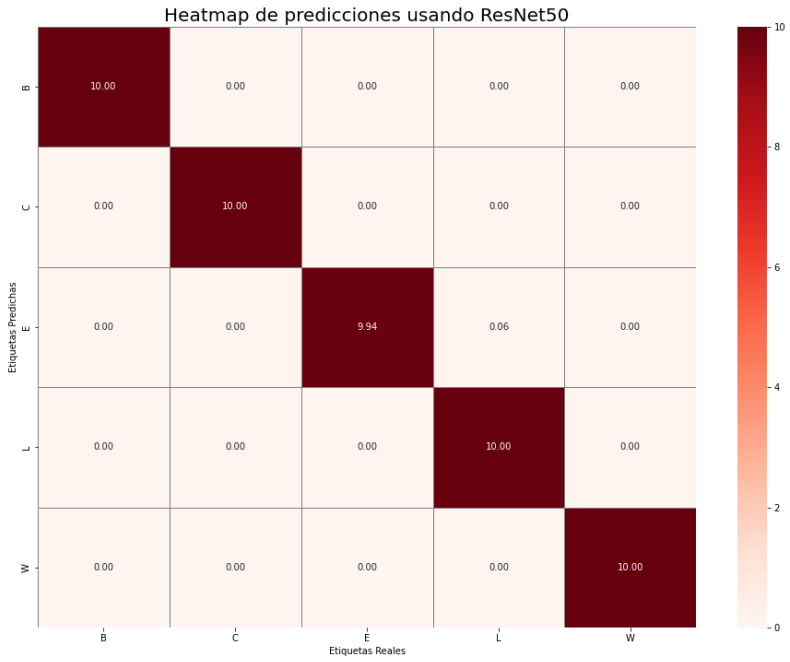


Figura 26: Mapa de calor de las predicciones de categoría ResNet50.

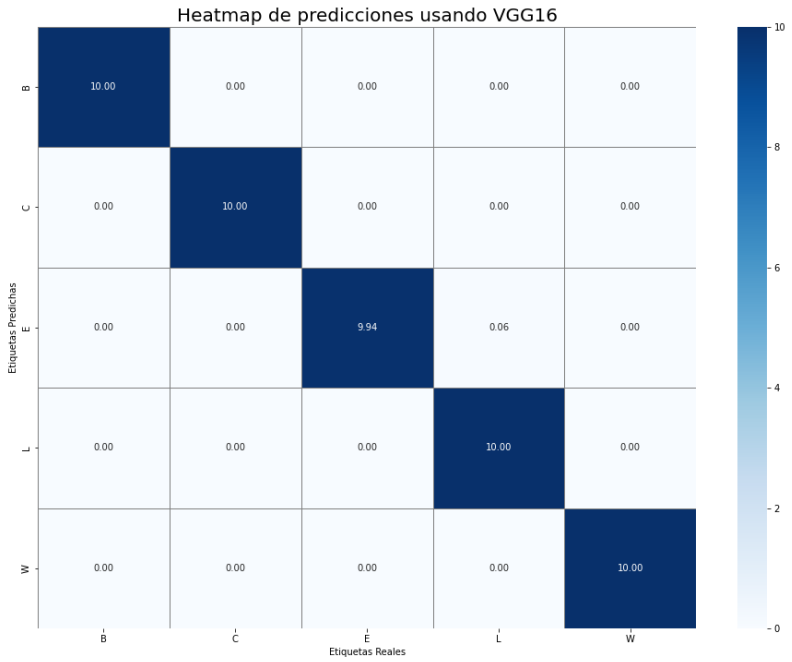


Figura 27: Mapa de calor de las predicciones de categoría VGG16.

A partir de los mapas de calor (**heatmap**) de predicciones presentados en la **Figura 26** y **Figura 27**, se pueden extraer las siguientes conclusiones:

1. **Precisión en la Predicción de Categorías**: La diagonal principal de cada matriz, que incluye las categorías **B**, **C**, **E**, **L** y **W**, muestra que las predicciones en términos de categorías son extremadamente precisas. Cada categoría se predice con una puntuación perfecta de **10 sobre 10**, lo que demuestra que los modelos son altamente efectivos en la clasificación de imágenes en sus categorías correspondientes.

2. **Errores Mínimos**: Las celdas fuera de la diagonal principal muestran errores mínimos en las predicciones. En particular, se observa una pequeña discrepancia en la categoría **E**, con una puntuación de **9.94**, y una puntuación residual de **0.06** en otra categoría. Este bajo nivel de error indica una alta precisión en la predicción de categorías.

3. **Consistencia en Predicciones**: El modelo demuestra una consistencia notable en sus predicciones, ya que todas las categorías **B**, **C**, **E**, **L** y **W** obtienen puntuaciones perfectas. Esto sugiere que el modelo es robusto y capaz de mantener una alta precisión en diversas categorías.

En resumen, los heatmap de predicciones muestran que los modelos alcanzan una precisión sobresaliente en la predicción de categorías, con errores mínimos y una consistencia destacada en todas las categorías evaluadas. Esto respalda la confiabilidad de los modelos en la clasificación precisa de imágenes en cada una de las categorías **B**, **C**, **E**, **L** y **W**.

5.3.3. Evaluación visual de las arquitecturas

Como se ha abordado en capítulos anteriores, el objetivo primordial de los modelos desarrollados consiste en predecir con precisión los puntos clave en el contexto de la Lengua de Señas Mexicana. Para alcanzar este objetivo, las diversas arquitecturas de modelos han sido entrenadas y evaluadas utilizando un conjunto de datos de prueba específico.
En el proceso de obtención de las coordenadas clave resultantes, se realiza una división clara en categorías principales que representan los dedos de una mano, a saber: Pulgar (**Thumb**), Índice (**Index**), Medio (**Middle**), Anular (**Ring**) y Meñique (**Little**). Estas categorías se diferencian mediante colores distintivos para facilitar su identificación y visualización en la imagen resultante. Es importante destacar que, dado que algunas señas pueden presentar dedos que no son visibles en su totalidad o partes de los dedos que puedan estar fuera del encuadre de la imagen, los puntos clave correspondientes a estas áreas faltantes se expresan de manera aproximada en la posición central de la imagen, considerando un tamaño de imagen de **200x200** pixeles como referencia.
Este enfoque de categorización y manejo de puntos clave es fundamental para comprender y visualizar de manera efectiva las predicciones de las arquitecturas de modelos en relación con la representación de las señas en la LSM.

5.3.3.1 Resultados de la letra B

Comenzando con los resultados de las predicciones relacionadas con la letra **B**, es importante destacar que esta letra se caracteriza por ser una de las más simples en cuanto a la representación de las coordenadas de los dedos en la LSM. Esto se debe a que los cuatro dedos principales (**Índice, Medio, Anular y Meñique**) generalmente se encuentran en una posición horizontal, y cada uno de estos dedos posee cuatro coordenadas clave, correspondientes a las articulaciones de dicho dedo y una coordenada adicional para representar la punta de este. Por otro lado, el pulgar se representa con tres coordenadas: dos para las articulaciones de este dedo y una tercera para la punta.
Uno de los resultados específicos que ilustra claramente cómo se representan estos puntos clave se presentan en la **Figura 28**. Esta figura proporciona una visualización concreta de la disposición y las coordenadas de los puntos clave para la letra **B**. Por otro lado, la **Figura 29** ejemplifica con claridad cómo las predicciones de coordenadas pueden experimentar desviaciones o alejarse del objetivo deseado. Esta representación visual pone de manifiesto situaciones en las que algunas coordenadas pueden no estar cercanas a los puntos clave esperados, lo que sugiere que, en ocasiones, el modelo puede cometer errores en la predicción de la posición exacta de los dedos en la señalización de la letra **B**.

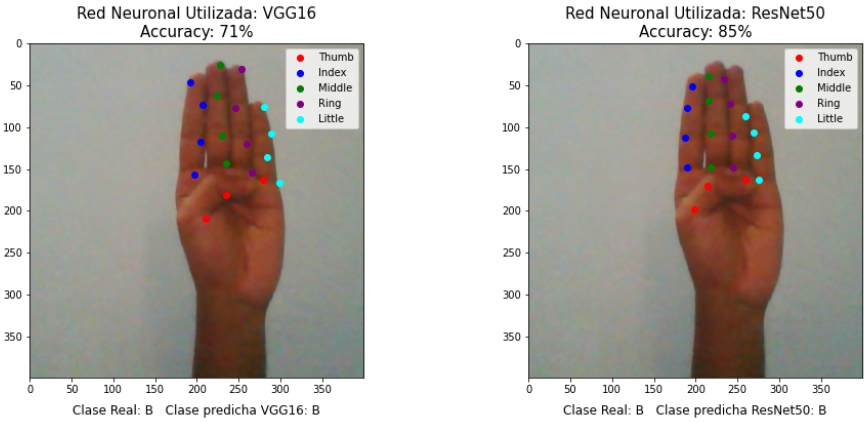


Figura 28: Mejor resultado que representan de manera correcta de los puntos clave en la seña **B**.Arquitectura VGG16 (**izquierda**) y ResNet50 (**derecha**).

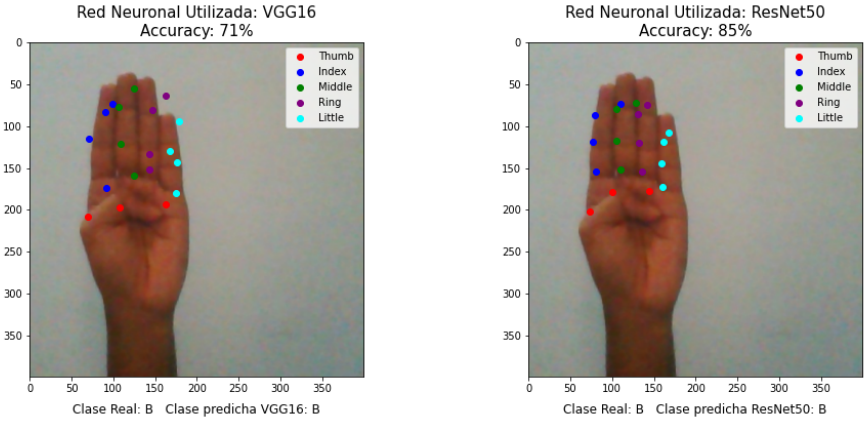


Figura 29: Resultado que representan como algunos puntos clave experimentan desviaciones en la seña **B**. Arquitectura VGG16 (**izquierda**) y ResNet50 (**derecha**).

Como se observa en la **Figura 28**, es evidente que los puntos clave predichos por ambas arquitecturas se asemejan significativamente a la forma de los dedos, con la arquitectura **ResNet50** mostrando una mayor precisión al seguir el patrón de los dedos. Sin embargo, en la **Figura 29**, se aprecia que varios puntos clave se aproximan a su ubicación esperada, pero otros se desvían considerablemente de su posición correcta, siendo esta discrepancia más notoria en el caso de la arquitectura **VGG16**.

5.3.3.2 Resultados de la letra C

El siguiente resultado a analizar y presentar se refiere a las pruebas realizadas con la **letra C**. Es relevante destacar que esta seña es peculiar, ya que su representación consiste en formar un puño con la mano, con los dedos extendidos y juntos, y el pulgar cruzado sobre la palma de la mano. Esta particularidad es importante porque, al estar la señal de lado, algunos dedos o partes de los dedos no son visibles al observarla. Los dedos que se muestran por completo son **el pulgar y el meñique**, representados por **3 y 4 puntos clave**, respectivamente (como se mencionó en la letra anterior).
Sin embargo, los dedos **anular y medio** se visualizan parcialmente, mostrando solo **2 y 1 punto clave relevantes**, respectivamente. Por último, es importante señalar que, dado que el dedo índice no es visible en la imagen, todos los puntos clave correspondientes a estos dedos se ubican en el centro de la imagen (**200x200 píxeles**), al igual que aquellos puntos que no son visibles en los demás dedos.
Al igual que en el caso anterior, la **Figura 30** proporciona una representación clara de cómo deben verse los puntos clave predichos para la letra **C**. Por otro lado, la **Figura 31** muestra desviaciones en las predicciones, en las que algunos puntos se acercan a la ubicación esperada, pero otros se alejan significativamente de su posición correcta.

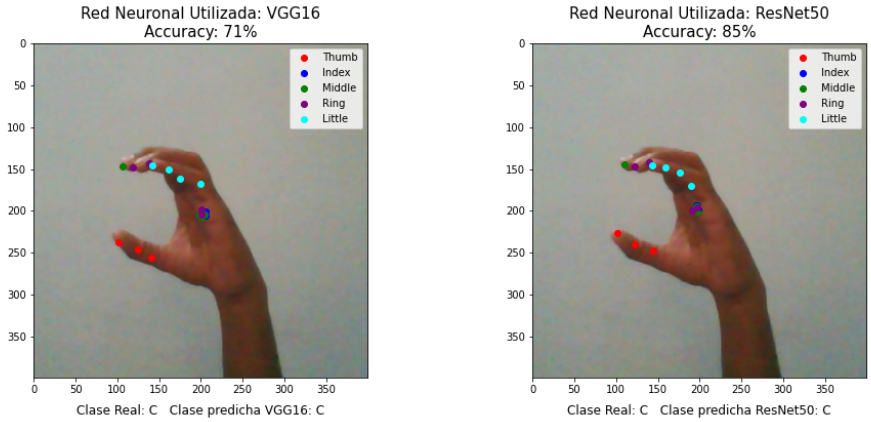


Figura 30: Mejor resultado que representan de manera correcta de los puntos clave en la seña **C**.Arquitectura VGG16 (**izquierda**) y ResNet50 (**derecha**).

Como se puede observar en la **Figura 30**, es evidente que los puntos clave predichos por ambas arquitecturas se asemejan a la forma de los dedos. De hecho, incluso los puntos que no son visibles en la imagen se representan correctamente en el centro de esta. Sin embargo, al examinar detenidamente los puntos clave, se aprecia que la arquitectura ResNet50 muestra mayor precisión en las predicciones.

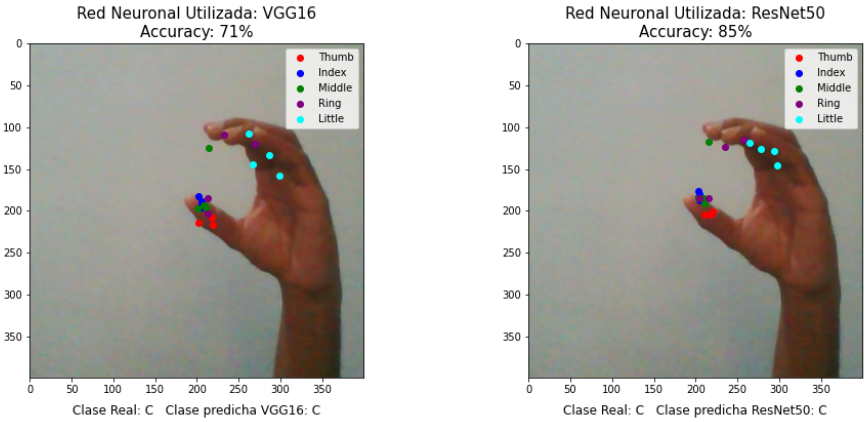


Figura 31: Resultado que representan como algunos puntos clave experimentan desviaciones en la seña **C**. Arquitectura VGG16 (**izquierda**) y ResNet50 (**derecha**)

Por otro lado, la **Figura 31** muestra desviaciones notables en ambas arquitecturas al predecir los puntos clave. Estas desviaciones pueden ser significativas en algunos casos, lo que resulta en predicciones de puntos clave que se encuentran fuera de la mano. Sin embargo, es importante destacar que la arquitectura **VGG16** exhibe un rendimiento inferior en términos de precisión de predicción en comparación con su contraparte, la arquitectura **ResNet50**. Las desviaciones son más pronunciadas en VGG16 en comparación con ResNet50.

5.3.3.3 Resultados de la letra E

La siguiente letra que analizamos es la letra **E**, la cual presenta características peculiares en su presentación. En esta seña, se coloca la mano en forma de puño, con los dedos presionando la palma de la mano y juntos, mientras que se mantiene el pulgar doblado y presionado contra los otros dedos. Estas características son relevantes al predecir los puntos clave, ya que, en todos los dedos, excepto el pulgar, una parte no es visible. Esto significa que solo tres de los cuatro puntos clave son visibles, lo que resulta en que los puntos faltantes se ubiquen en el centro de la imagen.
Los resultados de las pruebas, al igual que las anteriores, reflejan dos escenarios contrastantes. Por un lado, observamos predicciones que se acercan a la presentación precisa de la seña, logrando capturar con precisión la forma y la estructura deseada. Por otro lado, existe un escenario en el que las predicciones difieren considerablemente de la seña objetivo, lo que resulta en casos en los que los puntos clave se encuentran alejados de la forma original de la seña.
Como se puede observar en la **Figura 32** en esta ocasión, las predicciones realizadas por ambas arquitecturas se asemejan a la forma de la seña, acercándose a las posiciones que se esperan que sean, incluyendo aquellos puntos clave que deberían estar en el centro de la imagen. Sin embargo, en la **Figura 33** se presenta una situación contraria, donde solo unos pocos puntos clave se aproximan a las posiciones deseadas, resultando en una dispersión considerable y, en algunas ocasiones, alejándose de lo esperado.

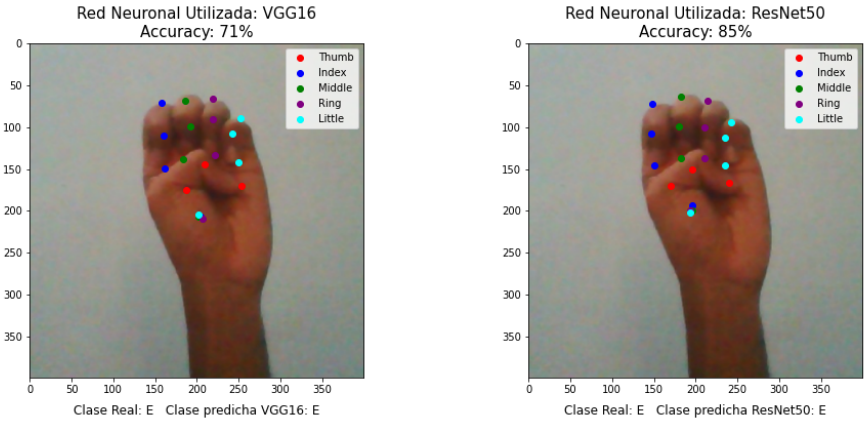


Figura 32: Mejor resultado que representan de manera correcta de los puntos clave en la seña **E**.Arquitectura VGG16 (**izquierda**) y ResNet50 (**derecha**).

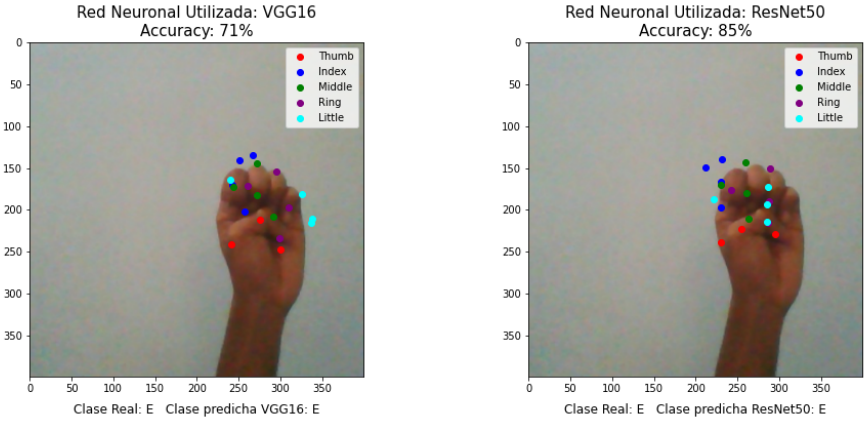


Figura 33: Resultado que representan como algunos puntos clave experimentan desviaciones en la seña **E**. Arquitectura VGG16 (**izquierda**) y ResNet50 (**derecha**).

Como se puede observar en la **Figura 32**, la arquitectura VGG16 demuestra una mayor precisión en la predicción de los puntos clave. Sin embargo, en la **Figura 33**, esta misma arquitectura presenta un rendimiento menos óptimo, ya que muestra una mayor dispersión en las predicciones. Por otro lado, la arquitectura ResNet50 en la **Figura 32** logra predicciones muy precisas, con solo algunas desviaciones en puntos clave, mientras que en la **Figura 33**, aunque aún presenta cierta dispersión en algunos puntos clave, supera en términos de precisión a la otra arquitectura.

5.3.3.4 Resultados de la letra L

La siguiente letra que se analiza en términos de resultados es la letra **L**, la cual se forma doblando los dedos medio, anular y menique hacia la palma de la mano, mientras se estiran los dedos pulgar e índice. Debido a que tres dedos están doblados hacia la palma de la mano, al igual que en señas anteriores, solo se visualizan tres puntos clave de estos dedos, mientras que el punto faltante se coloca en el centro de la imagen. Los puntos clave correspondientes al pulgar e índice se representan en su totalidad.
Como en los resultados anteriores, se observan dos escenarios contrastantes. Por un lado, se aprecian resultados en los que los puntos clave se asemejan mucho a la señal que se pretende predecir. La mayoría de estos puntos se predicen muy cerca o con pequeñas diferencias respecto a la posición real. Solo unos pocos puntos, excluyendo aquellos relacionados con partes no visibles, se desvían de la señal esperada. Por otro lado, existen casos en los que la mayoría de los puntos están desplazados, lo que resulta en que solo algunos puntos se acerquen a la posición correcta. Incluso los puntos clave relacionados con partes no visibles también presentan un desplazamiento considerable en estos casos.
Como se puede observar en la **Figura 34**, las predicciones realizadas por ambas arquitecturas arrojan resultados muy satisfactorios. En el caso de **ResNet50**, solo un punto del pulgar se encuentra ligeramente fuera de la señal, que corresponde al pulgar. Sin embargo, en ambas arquitecturas, los puntos clave siguen el patrón esperado de la señal. La arquitectura **VGG16** se acerca notablemente a las posiciones deseadas de los puntos clave.

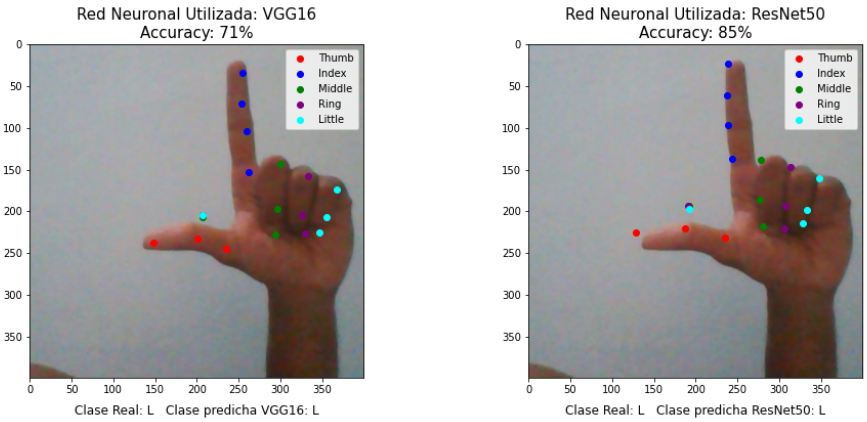


Figura 34: Mejor resultado que representan de manera correcta de los puntos clave en la seña **L**. Arquitectura VGG16 (**izquierda**) y ResNet50 (**derecha**).

Por otro lado, como se puede observar en la **Figura 35**, también existen predicciones en las cuales ambas arquitecturas muestran considerables desviaciones en los puntos clave, siendo más evidente en el caso de la arquitectura **VGG16**. En estas predicciones, no solo se encuentran puntos clave ubicados fuera de la señal, sino que también se presentan desplazamientos significativos en la posición de estos puntos. Esto incluye puntos clave que representan partes no visibles, así como puntos clave que se encuentran demasiado próximos entre sí cuando no deberían estarlo.

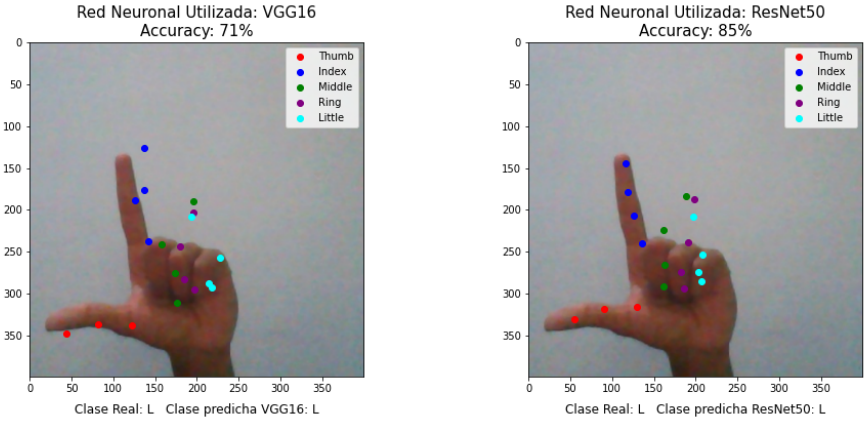


Figura 35: Resultado que representan como algunos puntos clave experimentan desviaciones en la seña **L**. Arquitectura VGG16 (**izquierda**) y ResNet50 (**derecha**).

Como se observó en ambos casos, ambas arquitecturas presentan aciertos y desviaciones al momento de predecir los puntos clave de la seña. Sin embargo, si nos basamos en ambas figuras (**Figura 34** y **Figura 35**), se evidencia que la arquitectura **VGG16** presenta errores más notables en algunas predicciones en comparación con **ResNet50**. Esto resulta en desviaciones considerablemente más notorias que respaldan la afirmación de que la arquitectura ResNet50 demuestra una mayor precisión en las predicciones de la seña.

5.3.3.5 Resultados de la letra W

Como última letra para presentar resultados, la letra **W**, al igual que las otras señas, presenta características únicas. Su representación física requiere que los dedos índice, medio y anular estén completamente extendidos y separados entre sí, lo que permite que los **4 puntos clave** de estos dedos sean visibles en la imagen. Sin embargo, la principal característica de esta señal es que el dedo pulgar y el meñique se encuentran doblados hacia la palma de la mano, con el dedo pulgar superpuesto sobre el dedo meñique. Esto significa que el dedo pulgar se representa con sus **3 puntos clave** correspondientes, mientras que los puntos clave del dedo meñique no son visibles y se representan en el centro de la imagen.

Al igual que en los resultados anteriores de las señas, se siguen observando tanto resultados positivos como negativos en ambas arquitecturas. Estos incluyen puntos clave que se asemejan bastante a la representación de la señal, siguiendo la forma deseada, así como predicciones en las cuales la mayoría de los puntos clave se encuentran fuera de la señal o siguiendo la forma del dedo que se desea predecir, pero con un desplazamiento considerable. Lo destacable es que los puntos clave correspondientes al dedo meñique se acercan mucho a lo mencionado anteriormente (centro de la imagen), lo que resulta en una gran precisión en ambas arquitecturas.
En la **Figura 36**, se pueden observar las mejores predicciones realizadas por ambas arquitecturas, y resultaron ser muy precisas ya que los puntos clave esperados se acercan a lo que se espera de estas predicciones. No obstante, se puede afirmar que la arquitectura **ResNet50** se destaca, ya que sus predicciones son aún más precisas, incluso en aquellos puntos que corresponden al dedo meñique, en comparación con la otra arquitectura.

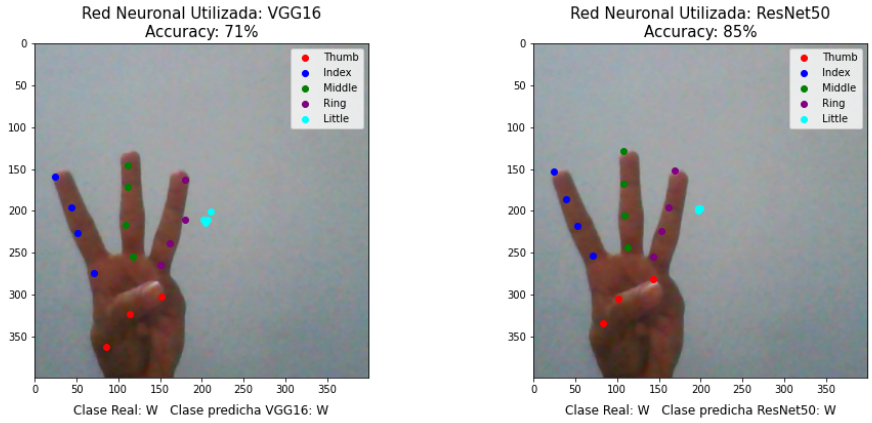


Figura 36: Mejor resultado que representan de manera correcta de los puntos clave en la seña **W**.Arquitectura VGG16 (**izquierda**) y ResNet50 (**derecha**).

En contraste, las predicciones que muestran desviaciones se presentan en la **Figura 37**, donde es evidente que la mayoría de los puntos clave están considerablemente desplazados, aunque aún siguen la forma esperada del dedo que se predice. No obstante, el hecho de que este desplazamiento lleve a los puntos clave fuera de la representación de la señal es un aspecto negativo de estas predicciones.
Es importante destacar que, si bien ambas arquitecturas muestran desviaciones, es en la arquitectura **VGG16** donde se observan estos casos con mayor frecuencia, incluso con puntos cercanos entre sí cuando no deberían estarlo. Además, en el caso del dedo meñique, las predicciones de **ResNet50** resultan más precisas que las de la otra arquitectura.

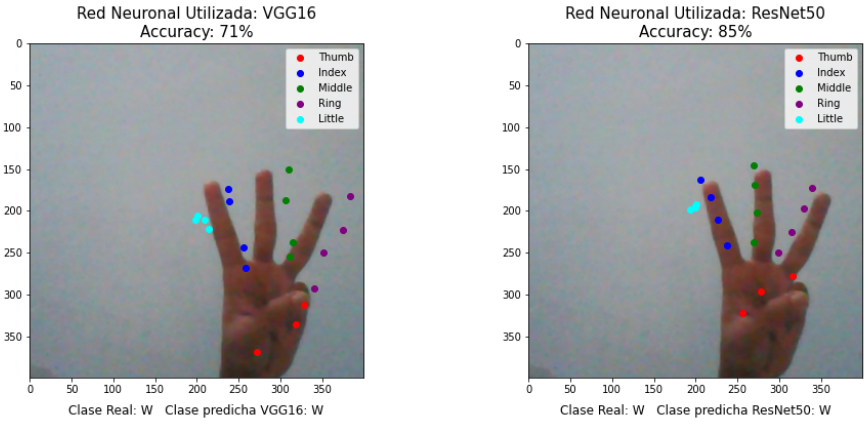


Figura 37: Resultado que representan como algunos puntos clave experimentan desviaciones en la seña **W**. Arquitectura VGG16 (**izquierda**) y ResNet50 (**derecha**).

Como se pudo observar en las figuras anteriores (**Figura 36** y **Figura 37**), ambas arquitecturas muestran tanto predicciones exitosas como desviaciones en sus resultados. No obstante, basándonos en los comentarios presentados, es evidente que la arquitectura ResNet50 sobresale en la precisión de las predicciones para esta señal en particular.

6. CONCLUSIONES

A lo largo de este estudio, hemos explorado detenidamente dos arquitecturas de redes neuronales convolucionales ampliamente reconocidas: **VGG16 y ResNet50**, en el contexto de la detección de puntos clave en la Lengua de Señas Mexicana. Nuestro objetivo principal es evaluar y comparar el rendimiento de estas arquitecturas en una serie de tareas críticas, como la precisión en coordenadas, la precisión en categorías, las pérdidas asociadas, el tiempo de entrenamiento y, en última instancia, su aplicabilidad en aplicaciones del mundo real.

En estas conclusiones, resumiré los hallazgos más relevantes de nuestro estudio, destacando cómo VGG16 y ResNet50 se desempeñan en cada una de las áreas mencionadas anteriormente. Siguiendo un esquema claro, comenzaré por recordar nuestros objetivos y metodologías, seguido de una revisión detallada de los resultados y su comparación. Luego, exploraré las fortalezas y debilidades de cada arquitectura. Además, reconoceré las limitaciones de nuestro estudio y proporcionaré sugerencias para futuras investigaciones en este campo en constante evolución.

A través de este proceso, arrojaré luz sobre cuál de estas arquitecturas podría considerarse la elección preferida en diferentes contextos y cómo pueden contribuir a mejorar la detección de puntos clave en un idioma de señas tan vital como el Mexicano.

6.1. Resumen de los Objetivos

Durante el desarrollo de este proyecto, he logrado satisfactoriamente el primer objetivo específico. Diseñé y desarrollé modelos de redes neuronales basados en arquitecturas previamente existentes, adaptados a las necesidades de la detección de la Lengua de Señas Mexicana. Estos modelos han demostrado ser **herramientas efectivas** en la comprensión y representación de los patrones visuales característicos de esta lengua de señas.

Para cumplir con el segundo objetivo específico, llevé a cabo la recopilación y creación de un conjunto de datos que contiene una amplia variedad de imágenes relacionadas con la LSM. Este conjunto de datos desempeñó un papel crucial en la formación y validación de los modelos de redes neuronales, proporcionando una base sólida para la evaluación y verificación de su rendimiento. Inicialmente, este conjunto de datos constaba de más de **4000 imágenes por categoría**, las cuales fueron reduciendo su cantidad a medida que avanzaba el proyecto.

El tercer objetivo específico se alcanzó al identificar y aplicar arquitecturas de redes neuronales que se ajustaban a los objetivos de la investigación. Seleccioné cuidadosamente aquellas arquitecturas que **demostraron ser eficaces** en tareas de detección de patrones visuales y las adapté para el reconocimiento preciso de la Lengua de Señas Mexicana.

Para el cuarto objetivo específico, realicé una evaluación exhaustiva del conjunto de datos en relación con los modelos de redes neuronales propuestos. Este proceso de validación permitió determinar la idoneidad y capacidad de los modelos para reconocer y comprender adecuadamente la LSM en diversas situaciones y contextos.
Finalmente, el quinto objetivo específico se centró en determinar la eficacia de los modelos de redes neuronales mediante una evaluación rigurosa utilizando el conjunto de datos recopilado. Los resultados de esta evaluación proporcionaron una visión clara del rendimiento de los modelos y permitieron tomar decisiones informadas sobre su utilidad en el reconocimiento preciso de la Lengua de Señas Mexicana.
En resumen, este proyecto ha logrado con éxito todos los objetivos específicos establecidos, desde el diseño y desarrollo de modelos de redes neuronales hasta la recopilación de datos, la identificación de arquitecturas adecuadas, la evaluación del conjunto de datos y la determinación de la eficacia de los modelos. Estos logros sientan las bases para futuras investigaciones y aplicaciones en el ámbito del reconocimiento de la Lengua de Señas Mexicana y otras áreas relacionadas.

6.2. Resumen de Metodologías

A lo largo de este estudio, he implementado y seguido una metodología rigurosa que ha sido esencial para alcanzar los objetivos establecidos en la detección de puntos clave en la Lengua de Señas Mexicana. Los pasos metodológicos cuidadosamente diseñados y ejecutados han permitido una evaluación exhaustiva de las arquitecturas de redes neuronales, VGG16 y ResNet50, en este contexto específico.
El primer paso de la metodología se centró en la obtención de los modelos de redes neuronales que fueran adecuados para los propósitos de la investigación. La **adaptación de modelos** previamente existentes a las necesidades de la detección de la LSM demostró ser exitosa, lo que resultó en la creación de herramientas efectivas para el reconocimiento preciso de patrones visuales.
La recopilación de un conjunto de imágenes diverso y representativo fue el segundo paso metodológico clave. Este conjunto de datos se convirtió en la **piedra angular** del proyecto, ya que proporcionó las imágenes necesarias para la formación y validación de los modelos de redes neuronales. La calidad y diversidad de los datos recopilados garantizaron resultados confiables y una evaluación exhaustiva.
El tercer paso metodológico implicó la identificación y aplicación de diversas arquitecturas de redes neuronales adecuadas para la investigación. La selección de VGG16 y ResNet50 se basó en su capacidad demostrada para tareas de **detección de patrones visuales**, lo que los convirtió en candidatos ideales para el reconocimiento de la Lengua de Señas Mexicana.
La cuarta etapa de la metodología se centró en la evaluación y validación del conjunto de datos en relación con los modelos de redes neuronales propuestos. Este proceso permitió **verificar la capacidad** de los modelos para comprender y representar adecuadamente la LSM en diversas condiciones y escenarios.

Finalmente, el quinto paso metodológico involucró una evaluación exhaustiva de los modelos de redes neuronales utilizando el conjunto de datos establecido. Los resultados de esta evaluación proporcionaron una visión clara del **rendimiento de los modelos**, destacando sus fortalezas y debilidades en el reconocimiento preciso de la Lengua de Señas Mexicana.
En conjunto, la metodología implementada ha sido fundamental para el éxito de este proyecto. Ha proporcionado un **marco sólido y estructurado** que ha permitido la consecución de los objetivos planteados y ha sentado las **bases futuras** para investigaciones relacionadas con el reconocimiento de la LSM y otras aplicaciones relacionadas. Esta metodología puede servir como guía para proyectos similares que busquen abordar desafíos de detección de patrones visuales en contextos específicos.

6.3. Resumen de los Hallazgos Clave

A lo largo de esta investigación, he realizado evaluaciones exhaustivas de dos arquitecturas de redes neuronales convolucionales ampliamente reconocidas, VGG16 y ResNet50, en el contexto de la detección de puntos clave en la Lengua de Señas Mexicana. Dichos hallazgos clave de esta investigación se pueden resumir de la siguiente manera:

1. **Rendimiento en Coordenadas**: Se observó que la arquitectura ResNet50 supera a VGG16 en términos de precisión en la predicción de coordenadas de puntos clave. ResNet50 logró una predicción significativamente mayor, lo que la convierte en una opción más adecuada para la localización precisa de puntos clave.
2. **Precisión en Coordenadas**: Ambas arquitecturas lograron una precisión del 100 % en la predicción de categorías de señas, lo que indica que ambas son efectivas para clasificar imágenes en las categorías correctas.
3. **Pérdida en Coordenadas**: ResNet50 mostró una pérdida en coordenadas significativamente menor en comparación con VGG16, lo que demuestra mayor precisión en la estimación de las coordenadas de los puntos clave.
4. **Tiempo de Entrenamiento**: ResNet50 requirió un tiempo de entrenamiento más largo en comparación con VGG16. Esto sugiere que ResNet50 es más intensiva en términos de tiempo de entrenamiento.
5. **Diferencias en las Predicciones de Señas**: Se observaron diferencias notables en las predicciones de señas entre las dos arquitecturas. ResNet50 demostró ser más precisa en la mayoría de los casos, mientras que VGG16 presentó desviaciones y errores más notables en algunas predicciones.
6. **Diversidad de Resultados**: Se encontró que ambos modelos VGG16 y ResNet50 proporcionaron tanto resultados positivos como negativos en términos de la precisión de las predicciones de puntos clave. Sin embargo, ResNet50 tendió a ofrecer resultados más precisos en general.

7. **Aplicabilidad en la Lengua de Señas Mexicana**: Ambas arquitecturas demostraron ser prometedoras para la detección de puntos clave en la Lengua de Señas Mexicana, lo que sugiere su aplicabilidad en aplicaciones del mundo real para personas con discapacidad auditiva.

En conjunto, estos hallazgos indican que **ResNet50** es la arquitectura preferida para tareas de **detección de puntos clave** en la Lengua de Señas Mexicana debido a su mayor precisión en coordenadas y menor pérdida en coordenadas. Sin embargo, se debe tener en cuenta que ResNet50 requiere **más tiempo de entrenamiento**, por lo cual, la elección entre ambas arquitecturas dependerá de las **necesidades específicas** del proyecto, equilibrando precisión y tiempo de entrenamiento. Además, estos resultados sugieren oportunidades para futuras investigaciones en la mejora de la detección de puntos clave en la LSM y otras aplicaciones relacionadas con la visión por computadora.

6.4. Comparación de Arquitecturas

A lo largo de este estudio exhaustivo, se ha explorado el rendimiento y la aplicabilidad de dos destacadas arquitecturas de redes neuronales convolucionales, VGG16 y ResNet50, en el contexto de la detección de puntos clave en la Lengua de Señas Mexicana. El objetivo ha sido evaluar y comparar meticulosamente estas arquitecturas en una serie de métricas críticas, que incluyen la precisión en coordenadas, la precisión en categorías, las pérdidas asociadas y el tiempo de entrenamiento.
En estas conclusiones, se resumen los hallazgos clave de nuestro estudio, destacando cómo VGG16 y ResNet50 se desempeñaron en cada una de las áreas mencionadas anteriormente. Dichas conclusiones se dividen en las siguientes comparaciones:

1. **Rendimiento en Coordenadas**: ResNet50 supera a VGG16 en términos de precisión en la predicción de coordenadas de puntos clave en la Lengua de Señas Mexicana. ResNet50 logró un impresionante 85 % de precisión, en comparación con el 71 % de VGG16. Esto indica que ResNet50 es más efectiva en la localización precisa de puntos clave en las imágenes.

2. **Precisión en Categorías**: Ambas arquitecturas demostraron una precisión del 100 % en la predicción de categorías de señas, lo que sugiere que ambas son altamente efectivas para clasificar las imágenes en las categorías correctas.

3. **Pérdida de Coordenadas**: ResNet50 exhibió una pérdida en coordenadas significativamente menor en comparación con VGG16. Esta menor pérdida indica que ResNet50 es más precisa en la estimación de las coordenadas de los puntos clave en las imágenes.

4. **Tiempos de Entrenamiento**: ResNet50 requirió un tiempo de entrenamiento más prolongado en comparación con VGG16. Esto indica que ResNet50 es más intensiva en términos de tiempo de entrenamiento, lo que podría ser un factor a considerar en proyectos con restricciones de tiempo.

5. **Diferencias en las Predicciones de Señas**: Se observaron diferencias notables en las predicciones de señas entre las dos arquitecturas. ResNet50 demostró ser más precisa en la mayoría de los casos, mientras que VGG16 presentó desviaciones y errores más notables en algunas predicciones.
6. **Aplicabilidad en la Lengua de Señas Mexicana**: Ambas arquitecturas mostraron promesa en la detección de puntos clave en la Lengua de Señas Mexicana, lo que sugiere su aplicabilidad en aplicaciones del mundo real para personas con discapacidad auditiva.

En resumen, ResNet50 supera a VGG16 en términos de precisión en coordenadas y pérdida en coordenadas, lo que la convierte en la opción preferida para tareas de detección de puntos clave en la Lengua de Señas Mexicana. Sin embargo, se debe tener en cuenta que ResNet50 también requiere más tiempo de entrenamiento. La elección entre ambas arquitecturas dependerá de las necesidades específicas del proyecto, equilibrando precisión y tiempo de entrenamiento. Estos hallazgos ofrecen una base sólida para futuras investigaciones y aplicaciones en el campo de la detección de puntos clave en la LSM y la visión por computadora en general.

6.5. Fortalezas y Debilidades

A lo largo de nuestro análisis comparativo de las arquitecturas VGG16 y ResNet50 en la detección de puntos clave en la Lengua de Señas Mexicana, hemos identificado diversas fortalezas y debilidades en ambas arquitecturas:

Fortalezas de VGG16:

1. **Precisión en Categorías**: VGG16 demostró una precisión del 100 % en la clasificación de categorías, lo que indica su eficacia en la identificación de las señas mexicanas correctas.
2. **Efectividad en Coordenadas**: En ciertos casos, VGG16 logró predicciones precisas de las coordenadas de puntos clave.
3. **Rapidez de Entrenamiento**: VGG16 generalmente requirió menos tiempo de entrenamiento en comparación con ResNet50.

Debilidades de VGG16:

1. **Precisión en Coordenadas**: VGG16 mostró una precisión menor en la predicción de coordenadas en comparación con ResNet50, con desviaciones más significativas en algunos casos.

2. **Pérdida de Coordenadas**: La pérdida en coordenadas de VGG16 fue ligeramente mayor que la de ResNet50, lo que indica una menor precisión en la estimación de puntos clave.
3. **Variabilidad en Predicciones**: En ciertos escenarios, VGG16 presentó una mayor variabilidad en las predicciones, con puntos clave más dispersos y errores notables.

Fortalezas de ResNet50:

1. **Mayor Precisión en Coordenadas**: ResNet50 logró una mayor precisión en la predicción de coordenadas en comparación con VGG16, con desviaciones menores en la mayoría de los casos.
2. **Pérdida en Coordenadas Baja**: La pérdida en coordenadas de ResNet50 fue significativamente menor, lo que indica una precisión mejorada en la estimación de puntos clave.
3. **Estabilidad en Predicciones**: ResNet50 presentó una mayor estabilidad en las predicciones, con puntos clave más consistentes y menores errores notables.

Debilidades de ResNet50:

1. **Tiempo de Entrenamiento Prolongado**: ResNet50 requirió un tiempo de entrenamiento significativamente más largo en comparación con VGG16, lo que puede ser una desventaja en aplicaciones en tiempo real.
2. **Variabilidad en Categorías**: En algunos casos, ResNet50 presentó una menor variabilidad en la clasificación de categorías, lo que podría afectar su capacidad para reconocer ciertas señas mexicanas más diversas.

En resumen, VGG16 destaca por su velocidad de entrenamiento y precisión en la clasificación de categorías, pero presenta desafíos en la precisión de coordenadas y la variabilidad en las predicciones. Por otro lado, ResNet50 muestra una mayor precisión en coordenadas y una pérdida en coordenadas más baja, pero a costa de un tiempo de entrenamiento prolongado. La elección entre estas arquitecturas dependerá de las necesidades específicas de la aplicación y el equilibrio entre precisión y tiempo de entrenamiento.

6.6. Impacto en la Aplicación Práctica

Durante mi investigación exhaustiva, he explorado detenidamente las arquitecturas de redes neuronales convolucionales VGG16 y ResNet50 en el contexto de la detección de puntos clave en la Lengua de Señas Mexicana. Los resultados obtenidos ofrecen una visión valiosa sobre la aplicabilidad de estas arquitecturas en la práctica y sus implicaciones en el reconocimiento preciso de la Lengua de Señas Mexicana.
Uno de los hallazgos más destacados es que ambas arquitecturas tienen el potencial de desempeñar un papel crucial en la detección de puntos clave en la lengua de señas mexicana. Sin embargo, es evidente que **ResNet50** emerge como mi opción preferida cuando priorizo la **precisión** en la predicción de coordenadas. Su capacidad para lograr una mayor precisión en la ubicación de **puntos clave**, en comparación con VGG16, sugiere que podría ser la elección más adecuada para aplicaciones donde la precisión es de suma importancia, como en aplicaciones médicas o de asistencia a personas con discapacidades auditivas.
En cuanto al impacto en la aplicación práctica, estos hallazgos tienen el potencial de influir significativamente en el desarrollo de tecnologías de reconocimiento de la Lengua de Señas Mexicana. La elección de la arquitectura de red neuronal adecuada dependerá de las necesidades específicas de la aplicación. Si se prioriza la precisión en la ubicación de puntos clave, ResNet50 es la elección preferida. Por otro lado, si se busca una alta precisión en la clasificación de categorías y velocidad de entrenamiento, VGG16 es una opción sólida.
En resumen, este estudio proporciona información esencial que puede guiar la selección de arquitecturas de redes neuronales en aplicaciones del mundo real relacionadas con la Lengua de Señas Mexicana. Estos hallazgos respaldan la idea de que la inteligencia artificial y el aprendizaje profundo pueden desempeñar un papel fundamental en la mejora de la comunicación y la accesibilidad para las personas con discapacidades auditivas, lo que hace que este campo de investigación sea aún más relevante y prometedor.

6.7. Limitaciones y Áreas para Investigaciones Futuras

Durante el curso de esta investigación, se han identificado algunas limitaciones que es importante reconocer. Estas limitaciones proporcionan oportunidades para futuras investigaciones y mejoras en el ámbito de la detección de puntos clave en la Lengua de Señas Mexicana. A continuación, destacaré estas limitaciones y sugeriré posibles áreas de investigación futura.

Limitaciones:

1. **Tamaño del conjunto de datos**: Una de las limitaciones clave de este estudio es el tamaño limitado del conjunto de datos disponible. Aunque se hizo un esfuerzo por recopilar un conjunto de datos representativo, la disponibilidad de imágenes de señas mexicanas fue un desafío. En futuras investigaciones, sería beneficioso contar con conjuntos de datos más grandes y diversificados para mejorar la generalización de los modelos.
2. **Variedad de las señas**: La Lengua de Señas Mexicana es rica y variada, con numerosos gestos y señas. Este estudio se centró en un conjunto específico de señas, pero hay muchas más que no se han explorado. Investigaciones futuras podrían abordar la variabilidad en las señas y cómo los modelos pueden adaptarse a gestos más complejos y menos comunes.
3. **Evaluación en condiciones del mundo real**: Aunque se realizaron evaluaciones exhaustivas en un entorno controlado, las condiciones del mundo real pueden ser más desafiantes debido a factores como la iluminación, el ruido y las variaciones en la forma en que se realizan las señas. Investigar cómo los modelos se desempeñan en situaciones del mundo real sería una extensión natural de este trabajo.

Áreas para Investigación Futura:

1. **Mejoras en la recopilación de datos**: Una investigación futura podría centrarse en la creación de conjuntos de datos más grandes y diversificados, posiblemente a través de la colaboración con comunidades de usuarios de Lengua de Señas Mexicana. Esto permitiría una mayor representación de señas y gestos.
2. **Modelos de aprendizaje profundo avanzados**: A medida que evoluciona la tecnología de aprendizaje profundo, se pueden investigar modelos más avanzados y complejos para mejorar aún más la precisión en la detección de puntos clave y la clasificación de categorías.
3. **Aplicaciones prácticas**: La implementación de sistemas basados en estos modelos en aplicaciones del mundo real, como dispositivos de comunicación para personas con discapacidades auditivas, es un campo prometedor para futuras investigaciones. Esto implicaría considerar cuestiones de accesibilidad y usabilidad.
4. **Adaptabilidad a otras lenguas de señas**: Si bien este estudio se centró en la Lengua de Señas Mexicana, los mismos enfoques pueden aplicarse a otras lenguas de señas en todo el mundo. Investigar la adaptabilidad de los modelos a diferentes lenguas de señas sería un área de interés.

En resumen, esta investigación sienta las bases para futuros avances en la detección de puntos clave en la Lengua de Señas Mexicana y abre puertas a una serie de oportunidades de investigación. A medida que la tecnología continúa avanzando, espero que estas limitaciones se aborden y que estas áreas para investigaciones futuras conduzcan a mejoras significativas en la accesibilidad y comunicación para la comunidad de personas con discapacidades auditivas.

6.8. Resumen General

Durante el transcurso de este proyecto, me embarqué en una investigación exhaustiva con el objetivo de analizar el desempeño de dos arquitecturas de redes neuronales convolucionales ampliamente conocidas, **VGG16 y ResNet50**, en el contexto de la detección de **puntos clave** en la Lengua de Señas Mexicana. Desarrollé e implementé una metodología sólida que incluyó desde la recopilación de datos hasta la implementación y evaluación de modelos, con el fin de alcanzar mis objetivos definidos previamente.
Uno de los hallazgos más significativos fue notar que la arquitectura **ResNet50 superó** de manera consistente a VGG16 en cuanto a la precisión en la predicción de coordenadas y la pérdida asociada. Este resultado señaló que ResNet50 **sobresalió** en la tarea de ubicar los puntos clave en las imágenes de señas. Además, ambas arquitecturas demostraron una eficacia sobresaliente al predecir las categorías de las imágenes, alcanzando una precisión del 100 %. Este hallazgo resalta la **idoneidad** de ambas arquitecturas para clasificar imágenes en categorías específicas.
Sin embargo, es importante tener en cuenta que la precisión del 100 % puede deberse al **sobreentrenamiento de los modelos** o a la limitada cantidad de imágenes utilizadas para su entrenamiento. Por lo tanto, en proyectos futuros, se debe considerar este aspecto y proponer nuevas soluciones para abordarlo.
En relación a las fortalezas, se constató que ResNet50 **destacó** en términos de **precisión y pérdida** en coordenadas, lo cual la posiciona como la **preferida** para las labores de detección de puntos clave. No obstante, también se notó que esta arquitectura demanda un **tiempo de entrenamiento** más prolongado en comparación con VGG16, lo que implica un balance entre precisión y eficiencia temporal.
Entre las limitaciones identificadas se encuentran el **tamaño reducido** del conjunto de datos y la variabilidad en los gestos. Además, se señaló la necesidad de adaptar los modelos a escenarios del mundo real y mejorar la recopilación de datos, aspectos que se presentan como áreas para futuras investigaciones.
En resumen, este proyecto **sienta bases sólidas** para futuros avances en la detección de puntos clave en la Lengua de Señas Mexicana y otras lenguas de señas, con el potencial de mejorar la accesibilidad y la comunicación para personas con discapacidades auditivas. La configuración del entrenamiento, incluyendo el orden de las imágenes, la tasa de aprendizaje y el número de iteraciones, debe ser adaptada a las necesidades específicas de cada proyecto, considerando la precisión, la pérdida y los requerimientos de tiempo. Este proyecto representa un avance significativo en la aplicación del aprendizaje profundo para abordar desafíos relevantes en la comunicación inclusiva.

7. REFERENCIAS

Referencias

[1] Asael Villanueva, *4 Frases en lengua de señas que todos deberiamos saber*, sep. de 2020.

[2] Dis-Capacidad, *Censo 2020: 16.5 % de la población en México son personas con discapacidad*, ene. de 2021. dirección: https://dis-capacidad.com/2021/01/30/censo-2020-16-5-de-la-poblacion-en-mexico-son-personas-con-discapacidad/.

[3] Organización Panamericana de la Salud, *Discapacidad*, 2020. dirección: https://www.paho.org/es/temas/discapacidad.

[4] Senado de la República, *Senado cambia término "capacidades diferentes" por "discapacidad", en Ley General de Desarrollo Social*, abr. de 2016. dirección: http://comunicacion.senado.gob.mx/index.php/informacion/boletines/27528-senado-cambia-termino-capacidades-diferentes-por-discapacidad-en-ley-general-de-desarrollo-social.html,.

[5] Jullian Christian, "Haciendo "hablar" a una historia muda. Surgimiento y consolidación de la comunidad sorda de Morelia," *SciELO*, vol. 39, mar. de 2018, ISSN: 2448-7554. dirección: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0185-39292018000100261#aff1.

[6] M. Chávez, *Gustavo Escobar, un lingüista que cambió la historia*, feb. de 2022. dirección: https://www.yotambien.mx/charla-sin-barreras/gustavo-escobar-un-linguista-que-cambio-la-historia/.

[7] L. Blasco, A. Llorente, I. Gil, T. Lari y N. Pianzola, *¿Cuáles son las letras menos usadas del español? — BBC Mundo*, oct. de 2019. dirección: https://www.youtube.com/watch?v=t-EjahwJLtY&t=8s.

[8] F. Sultana, A. Sufian y P. Dutta, "Advancements in Image Classification using Convolutional Neural Network," *Fourth International Conference on Research in Computational Intelligence and Communications Networks.*, mayo de 2019.

[9] N. Adaloglou, T. Chatzis, I. Papastratis et al., "A Comprehensive Study on Sign Language Recognition Methods," jul. de 2020.

[10] J. Hurwitz y D. Kirsch, *Machine Learning for dummies*, IBM Limited Edition, C. Burchfield, ed. Hoboken, NJ: John Wiley & Sons, Inc, 2018, págs. 3-4. dirección: https://www.ibm.com/downloads/cas/GB8ZMQZ3.

[11] K. Sanghvi, *Image Classification Techniques*, mayo de 2020. dirección: https://medium.com/analytics-vidhya/image-classification-techniques-83fd87011cac.

[12] T. Jain, *Basics of Image Classification Techniques in Machine Learning*, ago. de 2019. dirección: https://iq.opengenus.org/basics-of-machine-learning-image-classification-techniques/.

[13] J. Amat Rodrigo, *Máquinas de Vector Soporte (Support Vector Machines, SVMs)*, abr. de 2017. dirección: https://www.cienciadedatos.net/documentos/34_

maquinas_de_vector_soporte_support_vector_machines#Introducci%C3%B3n.

[14] M. Thangaraj y M. Sivakami, "Text classification techniques: A literature review," *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 13, págs. 117-135, 2018, ISSN: 15551237. DOI: 10.28945/4066.

[15] S. Qadir, *How to use SVMs for text classification.* dirección: https://www.educative.io/edpresso/how-to-use-svms-for-text-classification.

[16] C. Tison, N. Pourthié y J. C. Souyris, "Target recognition in SAR images with Support Vector Machines (SVM)," *International Geoscience and Remote Sensing Symposium (IGARSS)*, págs. 456-459, 2007. DOI: 10.1109/IGARSS.2007.4422829.

[17] C. Sukawattanavijit, J. Chen y H. Zhang, "GA-SVM Algorithm for Improving Land-Cover Classification Using SAR and Optical Remote Sensing Data," *IEEE Geoscience and Remote Sensing Letters*, vol. 14, n.º 3, págs. 284-288, mar. de 2017, ISSN: 1545598X. DOI: 10.1109/LGRS.2016.2628406.

[18] D. Internet Marketing, *Qué son los árboles de decisión y para qué sirven — Máxima Formación*, 2020. dirección: https://www.maximaformacion.es/blog-dat/que-son-los-arboles-de-decision-y-para-que-sirven/.

[19] Sitio Big Data, *Árbol de decisión en Machine Learning (Parte 1) - sitiobigdata.com*, dic. de 2019. dirección: https://sitiobigdata.com/2019/12/14/arbol-de-decision-en-machine-learning-parte-1/#.

[20] R. Sharma, *Guide to Decision Tree Algorithm: Applications, Pros & Cons & Example — upGrad blog*, oct. de 2020. dirección: https://www.upgrad.com/blog/guide-to-decision-tree-algorithm/#Decision_Tree_in_R_Example.

[21] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, n.º 2, pág. 1883, 2009, ISSN: 1941-6016. DOI: 10.4249/SCHOLARPEDIA.1883. dirección: http://www.scholarpedia.org/article/K-nearest_neighbor.

[22] HolyPython, *k Nearest Neighbor (kNN) History - HolyPython.com*, jul. de 2020. dirección: https://holypython.com/knn/k-nearest-neighbor-knn-history/.

[23] L. E. Peterson, *K-nearest neighbor - Scholarpedia*, nov. de 2013. dirección: http://www.scholarpedia.org/article/K-nearest_neighbor.

[24] L. Gonzalez, *K Vecinos más Cercanos - Teoría - Aprende IA*, jul. de 2019. dirección: https://aprendeia.com/k-vecinos-mas-cercanos-teoria-machine-learning/.

[25] Na8, *Algoritmo k-Nearest Neighbor — Aprende Machine Learning*, jul. de 2018. dirección: https://www.aprendemachinelearning.com/clasificar-con-k-nearest-neighbor-ejemplo-en-python/.

[26] IBM, *Usage of KNN - IBM Documentation*, dic. de 2021. dirección: https://www.ibm.com/docs/en/ias?topic=knn-usage.

[27] *El algoritmo K-NN y su importancia en el modelado de datos — Blog — España — Merkle*, sep. de 2020. dirección: https://www.merkleinc.com/es/es/blog/algoritmo-knn-modelado-datos.

[28] Na8, *Breve Historia de las Redes Neuronales Artificiales — Aprende Machine Learning*, sep. de 2017. dirección: https://www.aprendemachinelearning.com/breve-historia-de-las-redes-neuronales-artificiales/.

[29] IBM, *El modelo de redes neuronales - Documentación de IBM*, ago. de 2021. dirección: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model.

[30] IBM, *Perceptrón multicapa - Documentación de IBM*, jun. de 2021. dirección: https://www.ibm.com/docs/es/spss-statistics/SaaS?topic=networks-multilayer-perceptron.

[31] M. Saedd, *A Gentle Introduction To Sigmoid Function*, ago. de 2019. dirección: https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/.

[32] S. Kostadinov, *Understanding Backpropagation Algorithm — by Simeon Kostadinov — Towards Data Science*, ago. de 2019. dirección: https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd.

[33] J. McGonagle, G. Shaikouski, C. Williams, A. Hsu, J. Khlm y A. Miller, *Backpropagation — Brilliant Math & Science Wiki*, nov. de 2017. dirección: https://brilliant.org/wiki/backpropagation/.

[34] Y. LeCun, L. Bottou, Y. Bengio y P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, n.º 11, págs. 2278-2323, 1998, ISSN: 00189219. DOI: 10.1109/5.726791.

[35] B. Kumar, *Convolutional Neural Networks: A Brief History of their Evolution — by Brajesh Kumar — AppyHigh Blog — Medium*, ago. de 2021. dirección: https://medium.com/appyhigh-technology-blog/convolutional-neural-networks-a-brief-history-of-their-evolution-ee3405568597.

[36] ImageNet, *ImageNet*, mar. de 2021. dirección: https://image-net.org/index.php.

[37] O. Russakovsky, J. Deng, H. Su et al., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, n.º 3, págs. 211-252, dic. de 2015, ISSN: 15731405. DOI: 10.1007/S11263-015-0816-Y.

[38] J. Wei, *AlexNet: The Architecture that Challenged CNNs — by Jerry Wei — Towards Data Science*, jul. de 2019. dirección: https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951.

[39] A. Verma, *ZFNet: An Explanation of Paper with Code — by Abhishek Verma — Towards Data Science*, ago. de 2020. dirección: https://towardsdatascience.com/zfnet-an-explanation-of-paper-with-code-f1bd6752121d.

[40] K. Simonyan y A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, sep. de 2014. DOI: 10.48550/arxiv.1409.1556. dirección: https://arxiv.org/abs/1409.1556v6.

[41] R. Alake, *Deep Learning: GoogLeNet Explained — by Richmond Alake — Towards Data Science*, dic. de 2020. dirección: https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765.

[42] S. Das, *CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more... — by Siddharth Das — Analytics Vidhya — Medium*, nov. de 2017. dirección: https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5.

[43] K. He, X. Zhang, S. Ren y J. Sun, "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, págs. 770-778, dic. de 2015, ISSN: 10636919. DOI: 10.48550/arxiv.1512.03385. dirección: https://arxiv.org/abs/1512.03385v1.

[44] I. María y M. Vega, "LA ESCUELA NORMAL PARA PROFESORES DE SORDOMUDOS EN EL SIGLO XIX: UNA EXPERIENCIA SUI GENERIS DENTRO DE LA ESCUELA NACIONAL DE SORDOMUDOS," *XI Congreso Nacional de Investigación Educativa*, sep. de 2011. dirección: http://www.comie.org.mx/congreso/memoriaelectronica/v11/docs/area_09/2434.pdf.

[45] M. Cruz Aldrete, *La educación del sordo en México siglos XIX y XX: La Escuela Nacional de Sordomudos – Cultura Sorda*, 2009. dirección: https://cultura-sorda.org/la-educacion-del-sordo-en-mexico-siglos-xix-y-xx-la-escuela-nacional-de-sordomudos/.

[46] Centro de Documentación y Estudios SIIS Dokumentazio eta Ikerketa Zentroa-Fundación Eguía-Careaga Fundazioa, V. Gasteiz y Diputación Foral de Álava, "Buenas Practicas. en la atención a personas con discapacidad," *Vivir Mejor. Hacia una comunicación efectiva*, pág. 94, 2012.

[47] Instituto Nacional de Antropología e Historia, *Lengua de Señas Mexicana*, 2015. dirección: https://mexicana.cultura.gob.mx/es/repositorio/x2abesp3qm-4.

[48] Consejo Nacional para el Desarrollo y la Inclusión de las Personas con Discapacidad, *Día nacional de la Lengua de Señas Mexicana (LSM) — Consejo Nacional para el Desarrollo y la Inclusión de las Personas con Discapacidad — Gobierno — gob.mx*, jul. de 2019. dirección: https://www.gob.mx/conadis/articulos/dia-nacional-de-la-lengua-de-senas-mexicana-lsm-203888.

[49] C. O. Sosa Jiménez, H. V. Ríos Fígueroa, E. J. Rechy Ramírez, A. Marin Hernandez y A. L. González Cosío, "Real-Time Mexican Sign Language Recognition," *2017 IEEE International Autumn Meeting on Power, Electronics and Computin*, 2017.

[50] F. Solís, D. Martínez y O. Espinoza, "Automatic Mexican Sign Language Recognition Using Normalized Moments and Artificial Neural Networks," *Scientific Research Publishing*, 2016, ISSN: 1947-3931.

[51] G. García-Bautista, F. Trujillo-Romero y S. O. Caballero-Morales, "Mexican sign language recognition using kinect and data time warping algorithm," *2017 International Conference on Electronics, Communications and Computers, CONIELECOMP 2017*, abr. de 2017. DOI: 10.1109/CONIELECOMP.2017.7891832. dirección: https://ieeexplore.ieee.org/abstract/document/7891832.

[52] M. Tenney, *Microsoft Kinect – Hardware*, 2012. dirección: https://gmv.cast.uark.edu/scanning/hardware/microsoft-kinect-resourceshardware/.

[53] S.-O. Caballero-Morales y F. Trujillo-Romero, "3D Modeling of the Mexican Sign Language for a Speech-to-Sign Language System," vol. 17, n.º 4, págs. 593-608, 2013, ISSN: 1405-5546. DOI: 10.13053/CyS-17-4-2013-011.

[54] T. Starner, J. Weaver y A. Pentland, "Real-time american sign language recognition using desk and wearable computer based video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, n.º 12, págs. 1371-1375, 1998, ISSN: 01628828. DOI: 10.1109/34.735811.

[55] B. Copeland, *Artificial Intelligence*, ago. de 2022.

[56] J. McCarthy, "What is Artificial Intelligence?" *Stanford University*, nov. de 2004. dirección: http://www-formal.stanford.edu/jmc/%20https://borghese.di.unimi.it/Teaching/AdvancedIntelligentSystems/Old/IntelligentSystems_2008_2009/Old/IntelligentSystems_2005_2006/Documents/Symbolic/04_McCarthy_whatisai.pdf.

[57] Communicatios, *'Machine learning': ¿qué es y cómo funciona?* Nov. de 2019. dirección: https://www.bbva.com/es/machine-learning-que-es-y-como-funciona/.

[58] Real Academia Española, *Algoritmo*, 2022. dirección: https://dle.rae.es/algoritmo.

[59] Iberdrola, *'Machine Learning': definición, tipos y aplicaciones prácticas - Iberdrola*, 2022. dirección: https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico.

[60] IBM Cloud Education, *Supervised Learning*, ago. de 2020. dirección: https://www.ibm.com/cloud/learn/supervised-learning#toc-supervised-QVA1W1YW.

[61] IBM Cloud Education, *Unsupervised Learning*, sep. de 2020. dirección: https://www.ibm.com/cloud/learn/unsupervised-learning.

[62] J. M. Carew, *Reinforcement Learning*, mar. de 2021. dirección: https://www.techtarget.com/searchenterpriseai/definition/reinforcement-learning.

[63] IBM Cloud Education, *What is Deep Learning?* Mayo de 2020. dirección: https://www.ibm.com/cloud/learn/deep-learning.

[64] J. Brownlee, *What is Deep Learning?* Ago. de 2019. dirección: https://machinelearningmastery.com/what-is-deep-learning/.

[65] IBM Cloud Education, *¿Qué son las redes neuronales? - España — IBM*, ago. de 2020. dirección: https://www.ibm.com/es-es/cloud/learn/neural-networks.

[66] IBM Cloud Education, *Neural Networks*, ago. de 2020. dirección: https://www.ibm.com/cloud/learn/neural-networks.

[67] DataScientest, *Perceptron : qu'est-ce que c'est et à quoi ça sert ?* Mar. de 2022. dirección: https://datascientest.com/es/perceptron-que-es-y-para-que-sirve.

[68] The Genius Blog, *Basics of Multilayer Perceptron – A Simple Explanation of Multilayer Perceptron*, ene. de 2018. dirección: https://kindsonthegenius.com/blog/basics-of-multilayer-perceptron-a-simple-explanation-of-multilayer-perceptron/.

[69] D. Calvo, *Perceptrón Multicapa - Red Neuronal - Diego Calvo*, dic. de 2018. dirección: https://www.diegocalvo.es/perceptron-multicapa/.

[70] N. Seth, *Estimation of Neurons and Forward Propagation in Neural Net*, abr. de 2021. dirección: https://www.analyticsvidhya.com/blog/2021/04/estimation-of-neurons-and-forward-propagation-in-neural-net/.

[71] T. Wood, *Sigmoid Function*, ago. de 2021. dirección: https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function.

[72] M. Saeed, *A Gentle Introduction To Sigmoid Function*, ago. de 2021. dirección: https://machinelearningmastery.com/a-gentle-introduction-to-sigmoid-function/.

[73] M. Sotaquira, *La Función de Activación*, sep. de 2018. dirección: https://www.codificandobits.com/blog/funcion-de-activacion/#la-funci%C3%B3n-tangente-hiperb%C3%B3lica-tanh.

[74] J. Brownlee, *Softmax Activation Function with Python*, oct. de 2020. dirección: https://machinelearningmastery.com/softmax-activation-function-with-python/.

[75] D. Calvo, *Función de activación – Redes neuronales*, dic. de 2018. dirección: https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/.

[76] Data Science Team, *Función de activación Relu — Aprendizaje automático — DATA SCIENCE*, sep. de 2019. dirección: https://datascience.eu/es/aprendizaje-automatico/funcion-de-activacion-relu/.

[77] I. Gavilan, *Catálogo de componentes de redes neuronales (III): funciones de pérdida — Ignacio G.R. Gavilán*, mayo de 2020. dirección: https://ignaciogavilan.com/catalogo-de-componentes-de-redes-neuronales-iii-funciones-de-perdida/.

[78] J. Martinez, *Gradiente Descendiente para aprendizaje automático - IArtificial.net*, sep. de 2020. dirección: https://www.iartificial.net/gradiente-descendiente-para-aprendizaje-automatico/.

[79] K. Elijah, *Cross-Entropy Loss Function. A loss function used in most... — by Kiprono Elijah Koech — Towards Data Science*, oct. de 2020. dirección: https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e.

[80] I. Meza, *Descenso por gradiente (Gradient descent) — Ivan Vladimir Meza Ruiz*, nov. de 2016. dirección: https://turing.iimas.unam.mx/~ivanvladimir/posts/gradient_descent/.

[81] Jesus, *¿Qué es un Optimizador y Para Qué Se Usa en Deep Learning? - DataSmarts Español*, ene. de 2020. dirección: https://datasmarts.net/es/que-es-un-optimizador-y-para-que-se-usa-en-deep-learning/.

[82] A. Srinivasan, *Stochastic Gradient Descent — Clearly Explained !! — by Aishwarya V Srinivasan — Towards Data Science*, sep. de 2019. dirección: https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31.

[83] A. Rojano Aguilar, R. Salazar Moreno, L. Miranda y W. Ojeda Bustamante, "Algoritmo adama en la inteligencia artificial," en *Sexto Congreso Nacional de Riego, Drenaje y Biosistemas*, Hermosillo, Sonora: Universidad de Sonora,

jun. de 2021. dirección: https://www.riego.mx/congresos/comeii2021/files/ponencias/extenso/COMEII-21005.pdf.

[84] R. Alberto, *Redes Neuronales: Propagación hacia adelante y propagación hacia atrás — by Rubiales Alberto — Medium*, jun. de 2021. dirección: https://rubialesalberto.medium.com/redes-neuronales-propagaci%C3%B3n-hacia-adelante-y-propagaci%C3%B3n-hacia-atr%C3%A1s-4745c0fb6286.

[85] IBM Cloud Education, *Recurrent Neural Networks*, sep. de 2020. dirección: https://www.ibm.com/cloud/learn/recurrent-neural-networks.

[86] M. Isaksson, *Four Common Types of Neural Network Layers*, jun. de 2020. dirección: https://towardsdatascience.com/four-common-types-of-neural-network-layers-c0d3bb2a966c.

[87] A. Singh, *Layers in Neural network. Layers are logical collection of... — by Amit Singh Rathore — Nerd For Tech — Medium*, feb. de 2021. dirección: https://medium.com/nerd-for-tech/layers-in-neural-network-90d48a5a42fb.

[88] D. Unzueta, *Convolutional Layers vs Fully Connected Layers*, nov. de 2021. dirección: https://towardsdatascience.com/convolutional-layers-vs-fully-connected-layers-364f05ab460b.

[89] J. Bronwlee, *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*, abr. de 2019. dirección: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.

[90] J. Brownlee, *A Gentle Introduction to Dropout for Regularizing Deep Neural Networks*, ago. de 2019. dirección: https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/.

[91] M. Xiang, *Convolutions: Transposed and Deconvolution*, jul. de 2020. dirección: https://medium.com/@marsxiang/convolutions-transposed-and-deconvolution-6430c358a5b6.

[92] TechTarget Contributor, *Deconvolutional Networks (deconvolutional neural networks)*, abr. de 2018. dirección: https://www.techtarget.com/searchenterpriseai/definition/deconvolutional-networks-deconvolutional-neural-networks.

[93] Kevin, *Keras Flatten with a DNN example in Python*, ago. de 2021. dirección: https://neuralnetlab.com/keras-flatten-dnn-example/.

[94] J. Jeong, *The Most Intuitive and Easiest Guide for Convolutional Neural Network — by Jiwon Jeong — Towards Data Science*, ene. de 2019. dirección: https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480.

[95] T. Wood, *Softmax Function*, sep. de 2019. dirección: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer.

[96] Google Developers, *Multi-Class Neural Networks: Softmax*, jul. de 2022. dirección: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax.

[97] Tutor, *Redes neuronales profundas - Tipos y Características - Código Fuente*, abr. de 2019. dirección: https://www.codigofuente.org/redes-neuronales-profundas-tipos-caracteristicas/.

[98] R. Alake, *Understanding and Implementing LeNet-5 CNN Architecture (Deep Learning) — by Richmond Alake — Towards Data Science*, jun. de 2020. direc-

ción: https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342.

[99] J. Wei, *AlexNet: The Architecture that Challenged CNNs — by Jerry Wei — Towards Data Science*, jul. de 2019. dirección: https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951.

[100] Popular Networks, *VGG16 - Convolutional Network for Classification and Detection*, nov. de 2018. dirección: https://neurohive.io/en/popular-networks/vgg16/.

[101] Data Science Team, *Redes neuronales residuales - Lo que necesitas saber (ResNet) — DATA SCIENCE*, 2020. dirección: https://datascience.eu/es/aprendizaje-automatico/una-vision-general-de-resnet-y-sus-variantes/.

[102] S. Kumar, *Metrics to Evaluate your Classification Model to take the right decisions*, jul. de 2021. dirección: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/.

[103] d. Rodriguez, *¿Cuál es la diferencia entre parámetro e hiperparámetro?* Dic. de 2019. dirección: https://www.analyticslane.com/2019/12/16/cual-es-la-diferencia-entre-parametro-e-hiperparametro/.

[104] Numpy, *NumPy - About Us*, 2022. dirección: https://numpy.org/about/.

[105] Aprende con Alf, *La librería Numpy — Aprende con Alf*, mayo de 2022. dirección: https://aprendeconalf.es/docencia/python/manual/numpy/.

[106] Pandas, *Package overview — pandas 1.5.0 documentation*, 2021. dirección: https://pandas.pydata.org/docs/getting_started/overview.html.

[107] J. L. Chacon, *Introducción a Pandas, la librería de Python para trabajar con datos*, mar. de 2021. dirección: https://profile.es/blog/pandas-python/#Estructuras_de_datos_en_Pandas.

[108] A. Torres, *Aprendizaje automático en Python: Las principales características nuevas de Scikit-Learn 0.24 que debes saber.* Oct. de 2021. dirección: https://www.freecodecamp.org/espanol/news/aprendizaje-automatico-en-python-las-principales-caracteristicas-nuevas-de-scikit-learn-que-debes-saber/.

[109] Universidad de Alcalá, *Scikit-Learn, herramienta básica para el Data Science en Python*, 2022. dirección: https://www.master-data-scientist.com/scikit-learn-data-science/.

[110] P. Majumdar, *Scikit Learn - Introduction*, 2021. dirección: https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm.

[111] S. Yegulalp, *What is TensorFlow? The machine learning library explained — InfoWorld*, jun. de 2022. dirección: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html.

[112] Keras, *About Keras*, 2022. dirección: https://keras.io/about/.

[113] P. Sayak, *Keypoint Detection with Transfer Learning*, mayo de 2021. dirección: https://keras.io/examples/vision/keypoint_detection/.

[114] Srishilesh, *Understanding PASCAL VOC Dataset — Engineering Education (EngEd) Program — Section*, feb. de 2022. dirección: https://www.section.io/engineering-education/understanding-pascal-voc-dataset/.

[115] E. Hofesmann, *How to work with object detection datasets in COCO format — by Eric Hofesmann — Towards Data Science*, feb. de 2021. dirección: https://towardsdatascience.com/how-to-work-with-object-detection-datasets-in-coco-format-9bf4fb5848a4.

[116] V. Meel, *What is the COCO Dataset? What you need to know in 2022 - viso.ai*, 2022. dirección: https://viso.ai/computer-vision/coco-dataset/.

[117] T. Y. Lin, M. Maire, S. Belongie et al., "Microsoft COCO: Common objects in context," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, n.º PART 5, págs. 740-755, 2014, ISSN: 16113349. DOI: 10.1007/978-3-319-10602-1{_}48. dirección: https://viso.ai/computer-vision/coco-dataset/.

[118] P. Parker, *Labelling Images for Object Detection with LabelImg - Altis*, mayo de 2022. dirección: https://www.altisconsulting.com/insights/labelling-images-for-object-detection-with-labelimg/.

[119] K. Londhe, *American Sign Language: RGB image dataset of American sign language alphabets*, 2021. DOI: 10.34740/kaggle/dsv/2184214. dirección: https://doi.org/10.34740/KAGGLE/DSV/2184214%20https://www.kaggle.com/datasets/kapillondhe/american-sign-language.

[120] A. Thakur, *American Sign Language Dataset: American Sign Language Dataset for Image Classifcation*, 2019. dirección: https://www.kaggle.com/datasets/ayuraj/asl-dataset.