Universidad Nacional de Córdoba

FCEFyN

FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

Doctorado en Ciencias de la Ingeniería

Tesis Doctoral

RECONOCIMIENTO DE POSTURAS DE LAS MANOS PARA LA INTERACCIÓN NATURAL HUMANO – COMPUTADORA EN AMBIENTES INTELIGENTES A TRAVÉS DE CÁMARA RGB

Autor: **Ing. César Alejandro Osimani**

Director: **Dr. Roberto Gastón Araguás**

Noviembre 2023

RECONOCIMIENTO DE POSTURAS DE LAS MANOS PARA LA INTERACCIÓN NATURAL HUMANO – COMPUTADORA EN AMBIENTES INTELIGENTES A TRAVÉS DE CÁMARA RGB

por Ing. César A. Osimani

Director

Dr. R. Gastón Araguás - FRC, UTN

Comisión Asesora

Dr. Orlando Micolini - FCEFyN, UNC

Dr. Guillaume Hoffmann - FAMAF, UNC

Dra. Laura Vargas - FCEFyN, UNC

Esta Tesis fue enviada a la Facultad de Ciencias Exactas, Físicas y Naturales de la Universidad Nacional de Córdoba como requisito parcial para la obtención del grado académico de Doctor en Ciencias de la Ingeniería.

Córdoba, Argentina
Noviembre, 2023

unc Universidad Nacional de Córdoba

FCEFyN
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES

Universidad Nacional de Córdoba

Posgrado - Facultad de Ciencias Exactas, Físicas y Naturales

ACTA DE EXAMEN

Acta Nº: P10#56

Año Académico: 2023 **Fecha Examen:** 01/11/2023 **Ubicación:** Sede Unica - P10

Actividad: (P10-DI002) TESIS DOCTORADO EN CIENCIAS DE LA INGENIERIA

Turno: P10-(T209)_TESIS Doctorado Cs. de **Mesa:** 01-11-23 OSIMANI CESAR **Llamado:** Llamado del Turno P10-(T209)

Docentes: RULLONI, VALERIA SOLEDAD (Presidente) DELRIEUX, CLAUDIO AUGUSTO (Vocal) RODRIGUEZ GARCIA, DIEGO

Nº	Apellido y Nombre	Identificación	Instancia	Fecha	Nota	Resultado
1	OSIMANI, CESAR ALEJANDRO	DNI 26413920	Regular	01/11/2023	A (Aprobado)	Aprobado

Total: 1 **Evaluados:** 1 **Aprobados:** 1 **Desaprobados:** 0 **Ausentes:** 0

OBSERVACIONES:

Dedicado a mi madre y padre por darlo todo.

Agradecimientos

A mi director, el Dr. Gastón Araguás, por su guía y por ser fuente de conocimiento en este trayecto. Al Dr. Orlando Micolini y al Dr. Guillaume Hoffmann por sus enseñanzas para alcanzar los objetivos que se plantean en materia de investigación. Dedico un recuerdo especial, en memoria de la Dra. Laura Vargas, por su dedicación para conmigo. Son inspiración para mí.

A la Universidad Blas Pascal, mi hogar académico, que me aloja como investigador. Extendiendo mi gratitud al cuerpo docente, estudiantes, colegas y a la comunidad, cuyo apoyo ha sido esencial.

Al Dr. José Antonio Piedra Fernandez, al Dr. Juan Jesús Ojeda Castelo y al Dr. Luis Iribarne, de la Universidad de Almería, España, por formarme como investigador. El crecimiento que se logra al trabajar junto a ellos es inmenso.

Para quienes están muy cerca mío acompañando y apoyando en lo emocional, mi compañera Julieta; mi hermana y hermano, Eugenia y Hernán; mi amigo Gastón; mi amigo de trayectorias y aventuras educativas, Christian Seppey; y para quienes apuestan por el aprendizaje, y en compartirlo.

César Osimani
Centro de Investigación Aplicada y Desarrollo
en Informática y Telecomunicaciones (CIADE-IT)
Universidad Blas Pascal
Córdoba, 2023

Índice

Resumen

Palabras claves: Redes Neuronales Artificiales, Visión Artificial, Reconocimiento de gestos de las manos, Nube de puntos, Cámara digital RGB

El constante desarrollo y mejoras realizadas sobre los algoritmos de aprendizaje profundo ha motivado a realizar grandes inversiones de tiempo y dinero para la implementación de soluciones basadas en esta tecnología. Casos de éxito de nuevas empresas son cada vez más frecuentes en este campo, ya sea en el procesamiento del lenguaje natural, visión artificial o distintas áreas de la inteligencia artificial. La visión artificial, definida en pocas palabras, es la automatización de la vista humana. Actualmente podemos encontrar visión artificial en aplicaciones de diversos sectores, como agricultura, seguridad y vigilancia, automatización y control de calidad en las industrias, servicios de salud, logística, ciudades inteligentes, entre otros.

También hay momentos en que la visión artificial queda en un segundo plano y se limita a ser una herramienta de asistencia en las tareas que el humano realiza, por ejemplo, actuando como un medio para la interacción entre los humanos y las computadoras. En este sentido, el uso de la visión artificial para identificar los gestos de las manos ofrece una alternativa para el control de computadoras sin necesidad de tocar los periféricos o sistemas de mando, tal como el teclado, mouse o pantalla táctil.

Este trabajo presenta una solución que reconoce los gestos de la mano mediante el análisis de puntos de referencia tridimensionales ubicados en las articulaciones de la misma, los cuales definen su esqueleto. Estos puntos de referencia se extraen utilizando un modelo creado con técnicas de aprendizaje automático con el uso de una cámara web que permiten obtener 21 puntos de referencia distribuidos: uno en la muñeca y cuatro más en cada dedo. Cada punto de referencia es una estimación de una coordenada tridimensional (x, y, z) que corresponde a la ubicación (x, y) dentro de la imagen y la dimensión z es una estimación de la distancia hacia la cámara. Estos 21 puntos tridimensionales de cada mano detectada en las imágenes son los datos de entrada para una red neuronal profunda que permite identificar 9 gestos. Además del diseño de una arquitectura de red apropiada para esto, la creación de un dataset propio y el entrenamiento de la red, otra de las principales aportaciones de este trabajo es la implementación de un procesamiento de los datos previo a ingresar a la red. Este procesamiento es una normalización de los datos

y una transformación de los puntos de referencia, lo que mejora considerablemente el rendimiento del modelo. La evaluación del modelo propuesto entrega una tasa de aciertos del 99,87 % en las predicciones realizadas en el reconocimiento de los 9 gestos de la mano.

Finalmente, se realiza la implementación del modelo en una aplicación que se le da el nombre **Hand Controller**, la cual es una interfaz natural de usuario que permite tomar el control del teclado y mouse de una computadora a través de gestos (o secuencias de gestos) y desplazamientos de la mano.

Abstract

Keywords: Artificial Neural Network, Computer Vision, Hand Gesture Recognition, Point Cloud, RGB Digital Camera

The constant development and improvements made on deep learning algorithms have motivated large investments of time and money for implementations of solutions based on this technology. Successful cases of new companies are becoming more and more frequent in this field, whether in Natural Language Processing, Computer Vision or different areas of Artificial Intelligence. Computer Vision, defined in a nutshell, is the automation of human sight. Today we can find computer vision in applications in various sectors, such as agriculture, security and surveillance, automation and quality control in industries, health services, logistics, smart cities, among others.

Sometimes the computer vision remains in the background and is defined to being a tool to assist in the tasks that humans perform, for example, acting as a way for interaction between humans and computers. In this sense, the use of computer vision to identify hand gestures offers an alternative for control of computers without the need to touch peripherals or command systems such as keyboard, mouse or touch screen.

This work presents a solution that recognizes hand gestures by analyzing three-dimensional landmarks located at the joints of the hand, which define the skeleton of the hand. These landmarks are extracted using a model created with machine learning techniques with the use of a webcam to obtain 21 distributed landmarks: one on the wrist and four on each finger. Each landmark is an estimate of a three-dimensional coordinate (x, y, z) corresponding to the location (x, y) within the image and the z dimension is an estimate of the distance to camera. These 21 three-dimensional points of each hand detected are input data for a deep neural network to identify 9 gestures. In addition to the design of an appropriate network architecture for this, the creation of own dataset and the training of the network, another of main contributions of this work is the implementation of a processing of the data prior to entering the network. This processing is a normalization of the data and a transformation of the landmarks, which considerably improves the performance of the model. The evaluation of proposed model delivers a 99.87 % hit rate in the predictions made in the recognition of the 9 hand gestures.

Finally, the model is implemented in an application called Hand Controller, which is a Natural User Interface that allows to take control of keyboard and mouse of a computer through gestures (or sequences of gestures) and hand movements.

Riassunto

Keywords: Rete neurale artificiale, visione artificiale, Riconoscimento dei gesti della mano, Nuvola di punti, Fotocamera digitale RGB

Il costante sviluppo e miglioramento degli algoritmi di deep learning hanno motivato grandi investimenti di tempo e denaro nell'implementazione di soluzioni basate su questa tecnologia. Le start-up di successo sono sempre più frequenti in questo campo, che si tratti di elaborazione del linguaggio naturale, di visione artificiale o di diverse aree dell'intelligenza artificiale. La visione artificiale, in modo semplice, è l'automazione della vista umana. Oggi la visione artificiale trova applicazione in diversi settori, come l'agricoltura, la sicurezza e la sorveglianza, l'automazione e il controllo della qualità nelle industrie, i servizi sanitari, la logistica, le città intelligenti, tra gli altri.

A volte la visione artificiale rimane in secondo piano e si limita a essere uno strumento di assistenza per le attività svolte dall'uomo, ad esempio come mezzo di interazione tra l'uomo e il computer. In questo senso, l'uso della visione artificiale per identificare i gesti delle mani offre un'alternativa per controllare i computer senza dover toccare periferiche o sistemi di comando come la tastiera, il mouse o il touch screen.

Questo lavoro presenta una soluzione che riconosce i gesti della mano analizzando i punti di riferimento tridimensionali situati in corrispondenza delle articolazioni della mano, che definiscono lo scheletro della mano. Questi punti di riferimento vengono estratti utilizzando un modello creato con tecniche di apprendimento automatico con l'uso di una webcam che consente di ottenere 21 punti di riferimento distribuiti: uno al polso e altri quattro su ogni dito. Ogni punto di riferimento è una stima di una coordinata tridimensionale (x, y, z) corrispondente alla posizione (x, y) all'interno dell'immagine e la dimensione z è una stima della distanza dalla telecamera. Questi 21 punti tridimensionali di ciascuna mano rilevati nelle immagini sono i dati di input per una rete neurale profonda in grado di identificare 9 gesti. Oltre alla progettazione di un'architettura di rete appropriata, alla creazione di un set di dati e all'addestramento della rete, un altro dei principali contributi di questo lavoro è l'implementazione di un'elaborazione dei dati prima del loro ingresso nella rete. Questa elaborazione consiste in una normalizzazione dei dati e in una trasformazione dei benchmark, che migliora notevolmente le prestazioni del modello. La valutazione del modello proposto fornisce un tasso di successo del 99,87 % nelle previsioni fatte nel riconoscimento dei 9 gesti della mano.

Infine, il modello è stato implementato in un'applicazione chiamata Hand Controller, che è un'interfaccia utente naturale che permette di prendere il controllo della tastiera e del mouse di un computer attraverso gesti (o sequenze di gesti) e movimenti della mano.

Capítulo 1

Introducción

Capítulo 1

Introducción

Contenidos

Con la llegada de las computadoras personales a oficinas y hogares, surgió la necesidad de realizar mejoras continuas en la interacción entre las personas y las computadoras. Con ello apareció una nueva especialidad denominada Interacción Humano-Computadora (HCI: Human-Computer Interaction) para atender, principalmente, dos aspectos: Usabilidad y Experiencia de Usuario. Con la Usabilidad se busca que el usuario pueda obtener sus resultados cometiendo la menor cantidad de errores posibles, que le resulte sencillo de aprender y fácil de recordar. Con la Experiencia de Usuario se desean lograr aspectos agradables, atractivos, que motiven y otorguen satisfacción al usuario.

Inicialmente, el estudio de la Interacción Humano-Computadora se centró en las computadoras de escritorio, para luego incorporar también los teléfonos móviles y todo tipo de dispositivos inteligentes. Actualmente, este estudio abarca áreas como el diseño centrado en el usuario, el diseño de la interfaz de usuario y el diseño de la experiencia de usuario.

La tendencia es que las interfaces de usuario se integren en la vida cotidiana y se encuentren en cualquier lugar, no sólo en las pantallas. Se pretende un mundo en el cual se interactúe con las computadoras con todos los sentidos. El uso de micrófonos para reconocer el habla y las cámaras digitales para detectar los movimientos del cuerpo, son algunos ejemplos de cómo evoluciona la manera de interactuar con los dispositivos.

Para conseguir que esta interacción sea más parecida a la realizada entre personas, el uso de las manos al ser detectadas por una cámara digital es una manera atractiva para lograr una interacción más natural. Posiblemente el tacto, con el uso de los periféricos como el mouse, el teclado y las pantallas táctiles, siga siendo la forma de interacción más usada. Sin embargo, la interacción por voz sigue siendo adoptada por las compañías y las personas aprovechan esta tecnología como una interfaz manos libres en muchas situaciones y tareas cotidianas.

Una de las primeras maneras eficientes de interfaces de usuario es la denominada Interfaz por Línea de Comandos (CLI: Command-Line Interface) que permite la comunicación a través de una entrada de texto. Esto evolucionó hacia las Interfaces Gráficas de Usuario (GUI: Graphical User Interface) en la cual se utilizan elementos gráficos para presentar información y exponer las acciones disponibles en una pantalla. Actualmente, la interacción con las computadoras es mayormente con una interfaz gráfica de usuario a través del tacto con una pantalla táctil, en los casos de teléfonos inteligentes, o con mouse y teclado en computadoras de escritorio o portátiles. Asimismo la tendencia está orientada en el desarrollo de las denominadas Interfaces Naturales de Usuario (NUI: Natural User Interface) en las cuales los dispositivos de mando (tal como el mouse, teclado, pantalla táctil, entre otros) son reemplazados por el habla y los gestos, posturas y movimientos con el cuerpo, rostro y manos. Adicionalmente se pueden encontrar algunas subdivisiones en los tipos de interfaces y se pueden mencionar algunas: la Interfaz de Enfoque del Usuario (ZUI: Zooming User Interface) que se basa en una interfaz gráfica donde el usuario puede ajustar el tamaño del área de visión ampliando o reduciendo detalles de alguna región particular; la Interfaz de Usuario Orgánica (OUI: Organic User Interface) en la cual se involucra una interacción con pantallas flexibles manipulando

su forma física; la Interfaz mediante Voz del Usuario (VUI: Voice User Interface) que permite realizar el control a través del habla. En la Fig. 1.1 se visualiza la evolución de las interfaces considerando los tipos de interfaces más relevantes.

Figura 1.1: Evolución de las interfaces de usuario

Existe un esfuerzo constante por incorporar interacciones cada vez más naturales, y gracias a los avances de la visión artificial existen importantes trabajos que se enfocan en el diseño de interfaces basados en el reconocimiento de los gestos de las manos, rastreo del movimiento de los ojos, identificación de la postura del cuerpo y la cabeza, incluyendo también las expresiones faciales. La visión artificial extrae información del entorno procesando las imágenes que son obtenidas desde los sensores de imagen, principalmente en cámaras digitales y mayormente trabajando en el espacio de color RGB[1] en su etapa de captura de imágenes.

Es frecuente encontrar sistemas de visión artificial en entornos interactivos con el fin de lograr que los sistemas sean cada vez menos perceptibles en las tareas diarias de los usuarios. Es decir, que los usuarios puedan aprovechar el uso de la tecnología sin casi percibir que hacen uso de ella. Aquí aparece el concepto de Ambiente Inteligente, por medio del cual se busca que la tecnología participe en las actividades de las personas generando un entorno de interacción amigable y transparente. En Ambientes Inteligentes se pueden encontrar sistemas para el reconocimiento del habla, internet de las cosas, agentes inteligentes, sistemas embebidos, computación ubicua, conectividad, visión artificial, entre otros.

Al hablar de visión artificial también se está hablando de inteligencia artificial, ya que ambos términos se refieren a sistemas que buscan imitar la inteligencia humana para realizar tareas y utilizan datos recopilados para mejorar su desempeño iterativamente. Este tipo de sistemas de visión artificial, en particular, o sistemas de inteligencia artificial, en general, no son programados o configurados de manera explícita sino que aprenden y se forman a sí mismos. Aquí aparece el término Aprendizaje Automático (Machine Learning) y se puede mencionar también a las Redes Neuronales Artificiales como uno de los tipos de modelos computacionales utilizados para lograr el aprendizaje automático. Existen distintos métodos para realizar el aprendizaje, entre los cuales se encuentra el Aprendizaje Supervisado en el que se debe disponer de un grupo conocido de datos para el entrenamiento del modelo.

[1]RGB (Red, Green, Blue) es la composición del color en función de la intensidad de los colores primarios.

1.1. Problema de Investigación

Cuántas veces se ha visto en películas o leído en predicciones de la tecnología que se viene en los próximos años, información sobre el uso de las computadoras con las manos, manipulando elementos virtuales que se visualizan en el aire. Se pueden recordar algunas escenas en donde el personaje Tony Stark (Iron Man) manipula una pantalla virtual suspendida en el aire, haciendo zoom, desplazando elementos, y todo con una naturalidad fascinante; o en la película Minority Report en donde el personaje principal, interpretado por Tom Cruise, se coloca guantes para controlar la visualización de información en una pantalla. Todo esto lleva a esperar el momento en que esta tecnología alcance un desarrollo que permita una interacción tan natural y fácil de utilizar como se puede ver en estas películas.

En el estado del arte existe un notorio avance en la detección de las manos y, sin embargo, aún pueden faltar algunos años para alcanzar una interacción natural tal como la imaginamos. Existen dispositivos llamados escáneres LiDAR (Light Detection and Ranging) que facilitan la detección de las manos, y objetos en general, utilizando haces de luz que miden la distancia hacia los objetos que se encuentran en frente, permitiendo adquirir una elevada cantidad de puntos tridimensionales del entorno [Palieri et al., 2020]. Frecuentemente, a este grupo de puntos tridimensionales se los llama nube de puntos y su cantidad de puntos dependerá de las características del escáner. Los escáneres LiDAR llevan a interesantes aplicaciones, por ejemplo, construcción de modelos 3D a partir del escaneo de objetos reales [Zhang and Yang, 2019], identificación de objetos dentro de un entorno [Muhammad and Kim, 2020] o coches con conducción autónoma [Hsu et al., 2021]. También existen dispositivos que a la nube de puntos 3D obtenida por los escáneres LiDAR le agregan información de color adquirida por cámaras digitales RGB, dando lugar a las denominadas cámaras de profundidad o sensores D-RGB (Depth - Red Green Blue). Entre los más conocidos podemos mencionar a Kinect, Intel RealSense y Leap Motion Controller, los cuales se pueden encontrar en oficinas y hogares ya que son asequibles. Sin embargo, no son dispositivos de consumo frecuente, como sí lo son las cámaras RGB de bajo costo, como pueden ser las cámaras web integradas en una computadora portátil y una cámara web por USB. Con esto último, entra en juego una problemática relacionada a las posibilidades de acceder al uso de esta tecnología, ya sea por su costo o sea porque ya se encuentra integrada a las computadoras que la mayoría de los usuarios utilizan. El acceso a una cámara web está al alcance de la mayoría de los usuarios de computadoras, pero no sucede lo mismo con una cámara de profundidad o un sensor LiDAR.

En este contexto de visión artificial e interacción natural con los dispositivos, es posible considerar que existe una necesidad por nuevos aportes para la identificación de gestos con las manos con el fin de avanzar hacia una interacción cada vez más natural con las computadoras y destinado a usuarios que disponen de dispositivos de uso masivo[2]. Además de esta necesidad, existe una problemática en ciertos ambientes, lugares públicos o plantas industriales, en donde no se puede, o no se recomienda, utilizar los periféricos

[2]Dispositivos de uso masivo hace referencia a cámaras digitales RGB de menor costo, como las que vienen integradas a una computadora portátil, o una webcam por USB o las cámaras de la mayorías de los teléfonos inteligentes.

usuales, tal como teclado, mouse o pantallas táctiles. Uno de los principales motivos para no utilizar estos periféricos en lugares públicos o en una fábrica es el desgaste de los mismos por su uso descuidado. En la actualidad, un sistema con control a través de los gestos de las manos es enemigo de la facilidad de uso, pero tiene su razón de ser, ya que se justifica para muchas situaciones y entornos en donde no se pueden utilizar otros medios. No está destinado a entornos de oficinas, en donde los usuarios están sentados en un escritorio con posibilidad de usar el teclado y el mouse sino con aquellos entornos en donde estos periféricos no están disponibles, o porque conviene que no lo estén.

1.2. Definiciones pertinentes

A continuación se realizan las primeras definiciones para facilitar las explicaciones de la propuesta de esta tesis en los capítulos siguientes.

1.2.1. Visión artificial

En el ámbito cognitivo se puede definir que la percepción es la interpretación que hace el cerebro de los estímulos recibidos a través de los sentidos y afectada también por la experiencia previa. Haciendo referencia a la percepción visual, se trata del proceso para crear una realidad mediante la información lumínica captada por el ojo. Dicho esto, la Visión Artificial, o también llamada Visión Computacional o Visión por Computadora o, en inglés, Computer Vision[3], pretende dotar a las computadoras la capacidad de percibir la realidad tal como lo hacen los humanos. También se puede dar una definición que complementa la temática mencionando al Procesamiento de Imágenes Digitales, como aquellas técnicas para el reconocimiento, descripción y síntesis del contenido de una imagen digital.

El desarrollo de técnicas de visión artificial no es trivial y conlleva algunas dificultades. Para poner en evidencia alguna de ellas sin detenernos demasiado en los detalles, traemos a colación el refrán "Una imagen vale más que mil palabras" para visualizar algunas de las dificultades más comunes: ambigüedades, cambios de iluminación, cambios de escala, oclusiones y movimientos. En la Fig. 1.2 se muestran algunos ejemplos.

1.2.2. Red neuronal artificial

Las redes neuronales artificiales son modelos simplificados que emulan la manera en que el cerebro humano procesa la información. Es una forma de computación inspirada en modelos biológicos. Poseen, en general, un número elevado de unidades de procesamiento (neuronas) interconectadas y organizadas en capas. Normalmente se diferencian tres partes: una capa de entrada, una o varias capas ocultas y una capa de salida. En la Fig. 1.3 se muestra un red neuronal con una capa de entrada con 5 neuronas, una única capa oculta con 5 neuronas completamente conectada y una neurona en la capa de salida.

[3]En la temática de este trabajo existen muchos términos que se conocen principalmente en inglés, por lo que se hará uso cuando sea necesario.

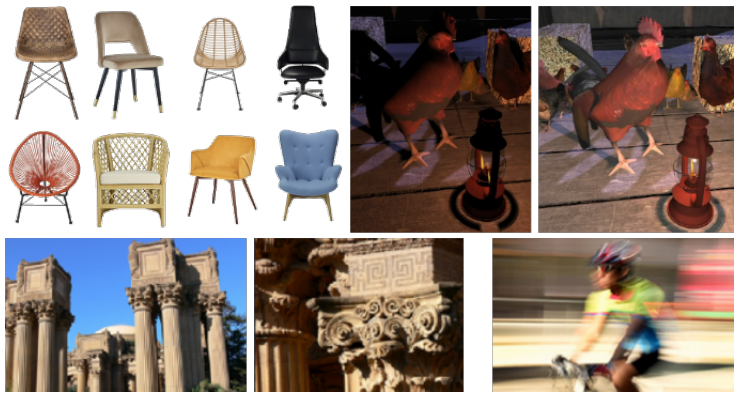


Figura 1.2: Ambigüedad, cambios de iluminación, cambios de escala y movimiento

Cuando cada neurona de una capa se conecta con todas las neuronas de la siguiente capa se la denomina capa completamente conectada o Fully Connected Layer.

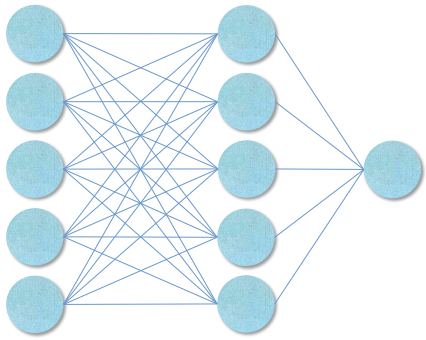


Figura 1.3: Red neuronal artificial

Las neuronas se interconectan con una variable de ponderación. Los datos de entrada se presentan en la capa de entrada y los valores se propagan desde cada neurona hacia las siguientes capas hasta llegar a la capa de salida presentando el resultado. Las redes neuronales pueden ser descriptas en función de su arquitectura, la cual hace referencia a características como: número de capas, número de neuronas en cada capa, tipo de neuronas, tipo de conexión entre neuronas y manera en que son entrenadas.

Una red neuronal debe ajustar las ponderaciones en base a un registro de datos conocido, llamado dataset. Con cada uno de estos registros van generando predicciones que se repiten muchas veces hasta alcanzar un criterio de parada. Al comenzar estas iteraciones, las ponderaciones son aleatorias y las predicciones generadas por la red pueden ser disparatadas. El aprendizaje de la red, es decir, el ajuste de las ponderaciones, se realiza continuamente a medida que se ingresan los datos conocidos de un dataset previamente confeccionado.

La red aprende examinando cada muestra del dataset, generando una predicción para cada muestra y realizando ajustes a las ponderaciones cuando realiza una predicción incorrecta. Este proceso se repite muchas veces y la red sigue mejorando sus predicciones hasta haber alcanzado uno o varios criterios de parada. El entrenamiento progresa y las predicciones generadas se van acercando cada vez más a los resultados conocidos. Cuando la red está entrenada, se obtiene un modelo que se puede utilizar para obtener predicciones de muestras para los cuales se desconoce el resultado, es decir, para muestras que no existen en el dataset con el cual fue entrenado el modelo.

1.2.3. Nube de puntos

Una nube de puntos es el nombre de fantasía usado para identificar a un conjunto de puntos en el espacio[4]. Una nube de puntos suele utilizarse para generar un modelo virtual (o modelo 3D) que permita lograr una representación fiel de una estructura del mundo real[5] y tener la posibilidad de trabajar digitalmente con dicho modelo. Una nube de puntos es el paso previo para generar un modelo 3D de objetos de pequeña escala (rostro, manos, mesas, sillas, botellas, etc.) y de gran escala (edificios, lugares históricos, fábricas, infraestructuras civiles, etc.). Hay diferentes formas de adquirir nubes de puntos que describan los objetos físicos que existen en un entorno, entre las más comunes se encuentra la utilización de escáneres LiDAR o escáneres láser (TLS: Terrestrial Laser Scanning) y cámaras de profundidad.

Para tener una idea de las características de un escáner láser industrial se puede presentar el escáner Leica RTC360 de la Fig. 1.4 que puede obtener hasta dos millones de puntos por segundo para crear una nube de puntos en color con alto rango dinámico (HDR: High Dynamic Range) de su entorno. Luego de obtenida la nube de puntos se requiere un proceso de acoplamiento por software para obtener el modelo 3D. Este proceso consiste en convertir los puntos en triángulos y polígonos para representar la superficie de los objetos escaneados.

Figura 1.4: Escáner láser 3D
Fuente: Leica Geosystems - `https://leica-geosystems.com`

[4]Nos referiremos al espacio tridimensional, pero el concepto es extensible a cualquier dimensión.
[5]Con la aparición de una manera alternativa de existencia, llamada virtualidad, debemos reforzar la referencia al mundo real, al verdadero, al de siempre.

1.3. Contribuciones y estructura de tesis

Una de las principales contribuciones de este trabajo consiste en el desarrollo de un modelo de aprendizaje profundo para reconocer 9 gestos de la mano mediante el análisis de una nube de puntos de la mano adquirida con cámaras RGB de bajo costo. Además, con su implementación en una aplicación para el control de computadoras en tiempo real con el diseño de secuencias de gestos y movimientos con las manos. La arquitectura de la red tiene una etapa previa de transformación y normalización de la nube de puntos que permite alcanzar muy buenas tasas de acierto en la clasificación de gestos.

Este aporte cuenta con la siguiente publicación de revista:

Enero 2023
Artículo: ***Point Cloud Deep Learning Solution for Hand Gesture Recognition***
Revista: International Journal of Interactive Multimedia and Artificial Intelligence
ISSN: 1989-1660, `http://dx.doi.org/10.9781/ijimai.2023.01.001`
Autores: César Osimani, Juan Jesus Ojeda-Castelo y José Antonio Piedra

En los próximos capítulos se explicarán los detalles para la creación e implementación de este modelo comenzando, en el capítulo 2, con una descripción de dispositivos para la adquisición de imágenes y algunas de las técnicas de visión artificial para resolver cuestiones de clasificación y detección de objetos. Relacionado a estas temáticas se realizaron aportes en las siguientes presentaciones en congresos:

Nov. 2017
Artículo: ***Seguimiento de la punta de la nariz y detección de la orientación de la cabeza con técnicas de visión artificial para una Interfaz Natural de Usuario***
Congreso: 5to Congreso Nacional de Ingeniería Informática / Sistemas de Información
pp. 915-919, ISSN: 2347-0372, `https://ria.utn.edu.ar/handle/20.500.12272/2677`
Autores: César Osimani y Emiliano Kohmann

Mayo 2015
Artículo: ***Detección de dientes en pacientes odontológicos mediante el análisis de imágenes en tiempo real***
Congreso: V Congreso de Matemática Aplicada, Computacional e Industrial
Vol. 5, pp. 543-546, `https://asamaci.org.ar/revista-maci`
Autores: César Osimani y Juan Carlos Ontiveros Neri

Nov. 2014
Artículo: ***Análisis y procesamiento de imágenes para la detección del contorno labial en pacientes de odontología***
Congreso: 2do Congreso Nacional de Ingeniería Informática / Sistemas de Información
pp. 147-152, ISSN: 2346-9927, `https://www.cesarosimani.com.ar/papers/contorno_labial.pdf`

Autor: César Osimani

El capítulo 3 aborda la primera etapa de la propuesta de esta tesis con la creación de un dataset propio, selección de los gestos de las manos y definición de una arquitectura de red neuronal. Algunos aportes relacionados a la detección de los gestos con las manos se presentaron en los siguientes congresos:

Abril 2017
Artículo: ***Hand Posture Recognition with standard webcam for Natural Interaction***
Congreso: 5th World Conference on Information Systems and Technologies
`https://link.springer.com/chapter/10.1007/978-3-319-56538-5_17`
Autores: César Osimani, José Antonio Piedra, Juan Jesus Ojeda-Castelo y Luis Iribarne

Junio 2016
Artículo: ***Hand Pose Estimation for Markerless Interaction with Augmented Reality Applications***
Congreso: ARGENCON Congreso Bienal de IEEE
`https://site.ieee.org/argencon/ediciones-pasadas/2016-06-15`
Autores: César Osimani y Emiliano Kohmann

El capitulo 4 presenta el proceso de entrenamiento y obtención de las métricas para la evaluación del modelo. En el capítulo 5 se muestra la implementación del modelo en una aplicación para el control del mouse y teclado de una computadora y, para finalizar, se realizan las conclusiones y comentarios finales sobre la integración de este medio de control en ambientes inteligentes y el diseño de interfaces naturales de usuario. Parte de estos aportes en las siguientes presentaciones:

Abril 2017
Artículo: ***Interfaz Natural de Usuario para el control de robot móvil con gestos faciales y movimientos de la cabeza usando cámara RGB***
Workshop: XIX Workshop de Investigadores en Ciencias de la Computación
pp. 403-407, ISBN: 978-987-42-5143-5, `http://sedici.unlp.edu.ar/handle/10915/61343`
Autores: César Osimani, Martín Salamero, Carlos Bartó y Marcos Lopez

Nov. 2015
Artículo: ***Simple método para Interacción Natural Humano-Computadora con movimientos del rostro***
Congreso: 3er Congreso Nacional de Ingeniería Informática / Sistemas de Información
`https://www.cesarosimani.com.ar/papers/simple_metodo.pdf`
Autores: César Osimani y Emiliano Kohmann

Finalmente, en el capítulo 6 se presenta una síntesis del desarrollo de esta tesis junto a las conclusiones generales y el planteo de trabajos futuros. La Figura 1.5 muestra un breve detalle del contenido de cada capítulo.

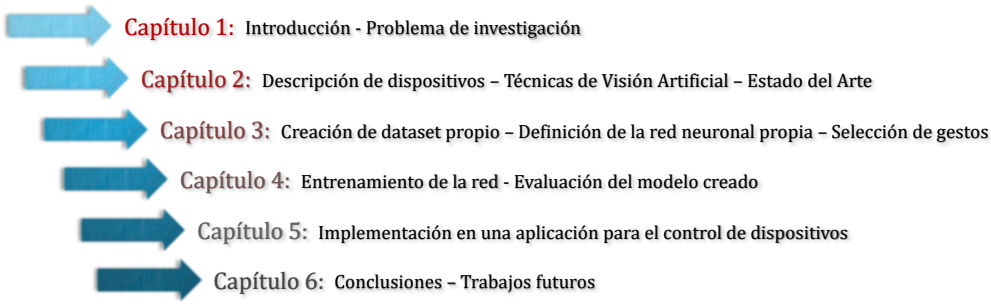


Figura 1.5: Capítulos de la tesis

Capítulo 2

Tecnologías y Estado del arte

Capítulo 2

Tecnologías y Estado del arte

Contenidos

Se describen en este capítulo las tecnologías[1] más determinantes para alcanzar los resultados de este proyecto. Se describirán no sólo las tecnologías que finalmente se utilizaron sino también aquellas que fueron descartadas pero que permitieron elaborar el estado del arte y así alcanzar los objetivos de esta tesis.

2.1. Periféricos de entrada

En esta sección se describirán algunos de los dispositivos (periféricos de entrada) involucrados en las técnicas de detección e identificación de las posturas de las manos. Si bien esta tesis se enfoca en hallar una solución de bajo coste mediante el uso de cámaras digitales RGB, también se evalúan dispositivos más costosos con el objetivo de explorar diversas técnicas del estado del arte y dirigirse hacia la opción más acertada.

2.1.1. Intel RealSense

Intel RealSense ofrece distintas soluciones de hardware y software de sencilla implementación, ideal para industrias, para desarrolladores, investigadores y educadores. La familia de productos de la serie D400 dispone de cámaras de profundidad con visión estereoscópica y cámara RGB. Ofrecen un kit de desarrollo de software (SDK: Software Development Kit) de código abierto y multiplataforma con aplicaciones ejemplo para obtener nube de puntos, realizar reconstrucción facial, detección de posturas del cuerpo humano, seguimiento de objetos, supresión de fondo, entre otros. Utiliza lenguaje de programación C++ e implementa otras bibliotecas de programación como OpenCV[2], ROS[3], OpenVINO[4], entre otros. En la Fig. 2.1 se visualiza el sensor RealSense D415 y en la Fig. 2.2 se observan conjuntamente la imagen color RGB y el mapa de profundidad, la cual es una imagen formada con colores que corresponden a las distintas distancias observadas por el sensor.

Figura 2.1: Cámara de profundidad RealSense D415.
Fuente: Intel RealSense - `https://www.intelrealsense.com`

[1] Con la palabra Tecnología se desea abarcar a los recursos técnicos, procedimientos, herramientas de software y hardware.
[2] OpenCV (Open Computer Vision) es una biblioteca desarrollada originalmente por Intel, de código abierto y ampliamente utilizada para visión artificial.
[3] ROS (Robot Operating System) es una herramienta de programación para el desarrollo de software para robots.
[4] OpenVINO (Open Visual Inference and Neural network Optimization) es un kit de desarrollo de código abierto para la utilización de modelos de deep learning en hardware Intel.

Figura 2.2: Imagen RGB y su mapa de profundidad.
Fuente: Intel RealSense - `https://www.intelrealsense.com`

Entre los trabajos relacionados se encuentra [Damindarov et al., 2021] en donde se utiliza una RealSense D435 para identificar los gestos de las manos a fin de controlar lo que se proyecta sobre una pizarra desde un proyector conectado a una computadora. También se pueden encontrar trabajos como [Ma and Yang, 2020] que proponen un sistema de escaneo con RealSense para la generación de modelos 3D de objetos pequeños como una manzana. Otra interesante aplicación de los sensores RealSense es para la exploración robótica autónoma, tal como [Bayer and Faigl, 2019] en la que un robot debe ser capaz de examinar un entorno desconocido y evaluar la transitabilidad.

2.1.2. Microsoft Kinect

El sensor Kinect proporciona imágenes RGB, mapa de profundidad y audio para ser utilizado a través de su SDK[5]. En la Fig. 2.3 se observan los sensores Kinect de los cuales las versiones 1 y 2 se encuentran discontinuadas pero fueron dispositivos ampliamente utilizados por grupos de investigación para el seguimiento de las partes del cuerpo e innovaciones en cuanto al escaneo 3D de entornos. Disponen de su SDK con implementaciones para el seguimiento del esqueleto humano, comandos por voz, reconstrucción de entornos mediante el escaneo, entre otros. Actualmente se encuentra disponible la versión llamada Azure Kinect.

Figura 2.3: Kinect v1, Kinect v2 y Azure Kinect.
Fuente: Kinect para Windows - `https://developer.microsoft.com/es-es/windows/kinect`

El sensor Kinect fue diseñado principalmente para brindar una nueva experiencia para los usuarios de la consola de videojuegos Xbox 360. No obstante, también fue adoptado por desarrolladores e investigadores para propuestas en otras disciplinas como

[5]Kinect para Windows SDK 2.0 - `https://developer.microsoft.com/es-es/windows/kinect`

en la robótica [Velayudhan and Gireeshkumar, 2015] para detección de obstáculos y navegación autónoma, para ámbitos educativos [Ono et al., 2020] con el diseño de una interfaz interactiva en la cual se controla con los gestos de las manos la proyección de material de estudio sobre el suelo de un aula, para rehabilitación de pacientes con algún trauma motriz en sus articulaciones [Lai et al., 2015] y diversas disciplinas más.

2.1.3. Leap Motion Controller

El sensor Leap Motion es una cámara de profundidad desarrollada principalmente para detección y seguimiento de las manos de los usuarios para que puedan interactuar de forma natural con las interfaces de usuario. Es pequeño, preciso y económico por lo que es atractivo para la creación de prototipos de AR/VR/XR[6] para investigación y desarrollo. Su zona interactiva se extiende hasta 60 cm desde el sensor y su SDK tiene técnicas implementadas para el análisis de nube de puntos que permiten identificar 27 elementos distintos de las manos, incluidos los huesos y articulaciones, y estimar su posición incluso cuando están ocluidos por otras partes de la mano. En la Fig. 2.4 se observa el sensor, que tiene pequeñas dimensiones (8 cm x 3 cm). En la web del producto[7] se pueden encontrar muy interesantes demostraciones relacionadas a interfaces sin los sistemas de mando habituales como mouse, teclado o pantalla táctil, sino a través de los gestos con las manos. Se encuentran implementaciones para check-in en hoteles, cajeros automáticos, manipulación de objetos virtuales en un entorno de realidad aumentada, entre otros.

Figura 2.4: Leap Motion Controller.
Fuente: Ultraleap - `https://www.ultraleap.com/product/leap-motion-controller`

Existen otros trabajos con Leap Motion como [Mittal et al., 2019] en el cual se identifican secuencias de gestos con las manos con el objetivo de interpretar la lengua de señas o trabajos como [Fadzli and Ismail, 2019] que genera un entorno de realidad aumentada en donde el usuario pueda modificar los objetos virtuales utilizando sus manos reales.

[6] AR (Augmented Reality) / VR (Virtual Reality) / XR (Mixed Reality)
[7] Galería de demos para Leap Motion - `https://gallery.leapmotion.com`

2.1.4. Cámaras digitales RGB

Los distintos dispositivos mencionados hasta aquí, tanto los escáneres LiDAR, escáneres láser o cámaras de profundidad, con sus correspondientes combinaciones con cámaras RGB, son muy interesantes para los proyectos de identificación de gestos de las manos con la gran ventaja de poder obtener una nube de puntos de precisión y con alta densidad, cosa que no se podría lograr, al menos por ahora, con las meras cámaras digitales RGB. Disponer de una nube de puntos densa y precisa permite utilizar herramientas de software y técnicas que facilitan la detección de las manos y sus correspondientes gestos, principalmente por disponer del SDK del fabricante del dispositivo con los algoritmos ya implementados para ello. Sin embargo, para este trabajo de tesis es muy importante el bajo coste de los dispositivos o, dentro de lo posible, que no se deba incurrir en un gasto extra. Es decir, que los usuarios ya dispongan del dispositivo con el cual se pueda realizar la tarea de detección de gestos en sus computadoras sin necesidad de adquirir un periférico extra o, en caso de necesitarlo, que sea económico adquirirlo. Esto abarca a las cámaras digitales RGB estándar, aquellas que ya vienen integradas en computadoras portátiles o aquellas cámaras comúnmente llamadas cámara web o webcam con conectividad por USB.

Una cámara RGB está compuesta por un sensor de imagen que mide la intensidad de luz dentro del espectro visible, es decir, el espectro que el ojo humano es capaz de ver. Además existen otro tipo de sensores que captan radiaciones que van más allá del espectro visible y son de mayor importancia en otras áreas como la agricultura. Para poder adquirir estas radiaciones se necesita un sensor o cámara multiespectral, pero no serán parte de este trabajo.

Las cámaras web tienen un lente, un sensor para captar la imagen y la electrónica para manejar la información sobre la imagen captada. Las tecnologías más utilizadas para la fabricación de los sensores de imagen de las cámaras digitales son CCD (Charge Coupled Device) y CMOS (Complementary Metal Oxide Semiconductor). Ambos tipos de sensores están formados por semiconductores de metal-óxido con celdas (llamados píxeles) distribuidas en forma de matriz, las cuales acumulan una carga eléctrica que depende de la cantidad de luz[8] que incida sobre dicha celda. Estos sensores convierten las cargas de las celdas en voltajes entregando una señal analógica en la salida, que será posteriormente digitalizada. Existen características como el rango dinámico que indica a partir de qué umbral mínimo son capaces de detectar energía lumínica y hasta qué umbral las celdas se saturan. Entre las diferencias entre las dos tecnologías se puede mencionar que los sensores CMOS tienen cada celda independiente por lo que la digitalización se realiza internamente sin necesidad de electrónica externa encargada para esta función, por lo que se logra mayor velocidad, mientras que en los sensores CCD las celdas no son independientes. Otras características de los sensores CMOS son sus precios son más bajos y menor consumo de energía, a costa de menor calidad en la imagen. Sin embargo, con los avances en los sensores CMOS se alcanzan muy buenos resultados por lo que es más común encontrar cámaras web con la tecnología CMOS.

Actualmente las cámaras web por USB o integradas en una computadora portátil

[8]Cantidad de luz o Energía lumínica es una porción de la energía transportada por la luz y que se manifiesta sobre un material.

son aptas para trabajar en resolución VGA (640×480 píxeles), HD (1280×720), Full HD (1920×1080) o Ultra HD (3840×2160) a 30 o 60 fotogramas por segundo (fps).

2.1.5. Elección de cámara

El sensor de imagen es parte fundamental en todo sistema de visión artificial y la elección del mismo depende de la aplicación y el resultado que se desee obtener. Las primeras preguntas que surgen cuando se debe seleccionar un sensor de imagen pueden ser: ¿Qué velocidad de fotogramas es requerida? ¿Qué nivel de detalle se desea? ¿En qué condiciones de iluminación se desea utilizar? Las respuestas a estas preguntas determinan algunas de las propiedades básicas de los sensores, como ser, la resolución, la cual tiene un papel clave para determinar el nivel de detalle deseado. En implementaciones para seguridad y vigilancia se necesita hacer énfasis en la sensibilidad y rango dinámico, para así hacer frente a posibles condiciones de baja iluminación. Para casos en que los objetos se encuentran en movimiento se puede exigir una mayor velocidad de fotogramas.

A continuación se presentan algunas propiedades importantes de un sensor de imagen:

- *Resolución:* La resolución determina el mínimo detalle a medir y se define por el número de celdas (o píxeles) en la superficie del sensor que están disponibles para la adquisición de imágenes. Por ejemplo, en un sensor de resolución VGA se tienen 640 columnas y 480 líneas de píxeles, haciendo un total de 307 200 píxeles, es decir, 0.3 megapíxeles (0.3 MP).

- *Sensibilidad:* La sensibilidad del sensor es importante para aquellas aplicaciones que deben trabajar con poca luz. La sensibilidad define el tiempo que se necesita para que los fotones que llegan al sensor se conviertan en electrones. Esta propiedad está influenciada por el ruido del sensor, que queda definida por la relación entre la señal generada por la imagen y el componente de ruido de la señal (SNR: Signal-to-Noise Ratio). Cuanto mayor sea la SNR, mayor será la calidad de las imágenes en aplicaciones con poca luz.

- *Rango dinámico:* La relación entre la mínima señal de la imagen (electrones de ruido) y el número máximo de electrones, sin que llegue a la saturación, define el rango dinámico. En otras palabras, define cuántos niveles de brillo diferentes se pueden generar.

- *Velocidad:* La velocidad del sensor (frame rate o tasa de fotogramas) determina el número de imágenes por segundo (fotogramas por segundo o fps) que el sensor puede leer electrónicamente. Una alta velocidad es importante en aplicaciones con objetos en movimiento, donde se pretende obtener imágenes sin desenfoque de movimiento. En general, los sensores CMOS alcanzan una mayor velocidad que los sensores CCD debido a su tecnología de lectura.

- *Obturación:* Los métodos de obturación determinan la manera en que los píxeles del sensor son expuestos y de qué manera se realiza su lectura. Dos métodos que normalmente se utilizan en los sensores CMOS son: Obturador Rodante

(Rolling Shutter) y Obturador Global (Global Shutter). El método de obturación rodante (usado en cámaras web) expone y lee los píxeles línea por línea, lo que suele presentar distorsiones con objetos en movimiento. La obturación global (usado en cámaras industriales) expone y lee simultáneamente todos los píxeles del sensor, logrando mayor nitidez sin distorsión con objetos en movimiento.

- *Color / Monocromático:* En la mayoría de los casos, un sensor de imagen monocromo es suficiente para las tareas típicas de visión artificial, como pueden ser, la detección de objetos, reconocimiento de textos o control de presencia. Para que un sensor proporcione una imagen color se aplican filtros píxel a píxel. Estos filtros se los llama máscara o mosaico de Bayer, que es un tipo de matriz de filtros, rojos, verdes y azules, que se colocan sobre el sensor para sólo hacer llegar información de uno de los colores. Con la interpolación de los píxeles adyacentes se forma la imagen color. Esta interpolación con el mosaico de Bayer reduce la precisión, por lo que un sensor de color sólo debe seleccionarse si la aplicación requiere información de color.

Las propiedades enumeradas anteriormente son útiles para realizar una primera lista de los posibles sensores. Los demás componentes de un sistema de visión y las condiciones en la que se deben desempeñar permitirán realizar una selección más acotada y precisa. En el mercado existe una gran cantidad de cámaras y sensores [9] que pueden ser utilizados en proyectos de visión artificial, ya sea para implementar en industrias, para realidad aumentada, para el diseño de interfaces humano-computadora, videollamadas, para propósitos artísticos, fotografía, creación de contenido multimedia, entre otros diversos rubros. La existencia de esta amplia variedad, cada cual con sus especificaciones, hace difícil la elección de la cámara adecuada y óptima para cada finalidad. En caso que los requerimientos de cámara no sean ni muy específicos ni de elevada calidad, la mayoría de las cámaras entregarán buenos resultados. Para colocar algunos ejemplos se puede considerar la compañía Teledyne FLIR[10] y sus propuestas en cuanto a las cámaras y sensores de imagen. Esta compañía presenta una tabla, a la cual llama "tabla periódica de sensores", que facilita la comparación de cámaras en base a su resolución, tamaño de píxeles, velocidad de fotogramas y formatos ópticos. Es un recurso útil para que, rápidamente, se puedan seleccionar algunos de los modelos para cada proyecto. Esta tabla, que se muestra en la Fig. 2.5, incluye información sobre los sensores de imagen de los distintos fabricantes que son utilizados en las cámaras de Teledyne FLIR. Cuando el sistema de visión que se está diseñando tiene características específicas y de alto rendimiento, esta tabla es de utilidad. Además se debe saber que, para disponer de estos productos, hay que realizar una considerable inversión económica tanto en la cámara como en todo el hardware y software que da soporte para su funcionamiento.

[9] Aquí se puede hacer una distinción entre sensor y cámara. Se llamará sensor de imagen al encargado de convertir los fotones en electrones, realizar la digitalización y generar la información para la imagen digital. Por otro lado, se utilizará la palabra cámara para identificar al dispositivo formado por el sensor de imagen, la óptica, la electrónica y el armazón.

[10] Teledyne FLIR diseña, desarrolla, fabrica y comercializa tecnologías relacionadas a sistemas termográficos, generación de imágenes con luz visible, análisis de vídeo, detección, medición y diagnóstico. `https://www.flir.com`

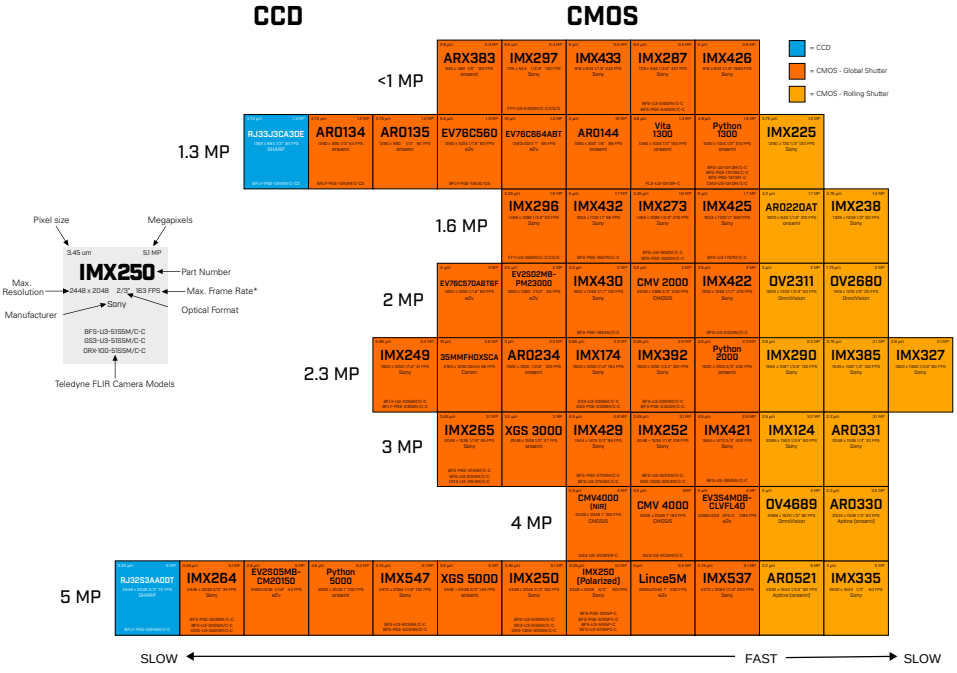


Figura 2.5: Tabla periódica de sensores.
Fuente: Teledyne - `http://www.flir.com/mv`

Además de las características propias de los sensores de imagen, se encuentran distintos tipos de cámaras que hacen uso de dichos sensores, que son construidas para desempeñarse en distintos ámbitos. Algunos tipos de cámaras disponibles se describen a continuación:

- *Cámaras matriciales*: Elaboran una imagen con un sensor que cubre una matriz de píxeles con distintas relaciones de aspecto, por ejemplo, 4:3 o 16:9.
- *Cámaras de alta velocidad*: Cámaras que capturan gran cantidad de fotogramas por segundo.
- *Cámaras lineales*: Generan una imagen línea a línea, desplazando la propia cámara o el objeto que se desea capturar.
- *Cámaras térmicas o infrarrojas*: Trabajan más allá del espectro de luz visible por el ojo humano. Permite formar imágenes en función de la temperatura de los objetos. En general, los objetos con mayor temperatura emiten más radiación infrarroja que otros de menor temperatura.

- *Cámaras multiespectrales*: Capturan y procesan una mayor cantidad de longitudes de onda, permitiendo diferenciar entre distintos tipos de materiales, lo que proporciona una gran recopilación de información a lo largo del espectro electromagnético.

- *Cámaras estereoscópicas*: De la misma manera que funciona la visión humana, este tipo de cámaras permiten obtener dos imágenes con las cuáles se puede reconstruir la escena tridimensional.

- *Cámaras 360*: Construyen una visión panorámica del entorno.

Entre las finalidades de este proyecto de tesis se encuentra la utilización de cámaras de bajo coste y uso masivo, con el objetivo de llegar a mayor cantidad de usuarios. Por ello, la mayoría de las cámaras y sensores comerciales quedan descartadas por su costo y porque sería un sobredimensionado de características para las aplicaciones a las que se orienta esta tesis. En consecuencia, luego de la exploración del mercado de cámaras y sensores de imagen, el análisis de los parámetros principales que definen sus prestaciones y el ámbito para la que están diseñadas, se puede determinar un abanico posible de cámaras para este proyecto. En la Fig. 2.6 se muestra una tabla con las características que pueden ser encontradas en las especificaciones de los sensores CMOS de algunas cámaras comerciales[11] y sensores de imagen integrados en computadoras portátiles. Notar que, en general, los fabricantes de sensores de imagen no fabrican las cámaras, sino que existen fabricantes de cámaras que seleccionan y utilizan sensores de imagen de terceros.

Fabricante sensor	Modelo de Sensor	Pixel Size	Array Size	Resolución & fps	Producto final	Precio
Uctronics	NT99141	3 µm	1280 x 720	640x360 → 60fps 1280x720 → 30fps	Cámara web Genius FaceCam 1000X	U$S 8
Sony	IMX307	2,9 µm	1920 x 1080	1920×1080 → 60fps	Computadora portátil Dell Inspiron 5502	U$S 12
ams OSRAM	Mira030	3,2 µm	640 x 480	640×480 → 180fps	Kit de desarrollo NVIDIA Jetson Nano	U$S 14
OmniVision	OH02A1S	1,4 µm	1920 x 1080	1280x720 → 90fps 1920x1080 → 60fps		U$S 28
Aptina	MI5100	2,2 µm	2592 x 1944	2592x1944 → 15fps 1920x1080 → 30fps		U$S 55
IADIY	IMX179	1,4 µm	3264 x 2448	3264x2448 → 15fps 1024x768 → 30fps		U$S 86
Sony	FSM-IMX415C	1,45 µm	3864 x 2176	3864x2176 → 90fps		U$S 192

Figura 2.6: Algunos sensores disponibles en el mercado

[11] Los precios colocados en esta tabla son una referencia aproximada en dólares obtenidos de diversas plataformas dedicadas al comercio electrónico.

A continuación se presenta un listado de los fabricantes de sensores de imagen que pueden ser útiles para implementaciones en visión artificial:

- *ams OSRAM*: Ofrece tecnología para la detección, iluminación y visualización. En su cartera de productos se encuentran LEDs, lasers, fotodetectores y sensores de imagen. https://ams-osram.com
- *IADIY*: Desarrolla y fabrica productos para optoelectrónica, detección robótica y sensores de imagen. https://www.iadiy.com
- *Aptina Imaging Corporation*: Aptina fue una empresa dedicada al desarrollo de sensores de imagen que, en el año 2014, fue adquirida por la empresa onsemi. https://www.onsemi.com/products/sensors/image-sensors
- *Sony Semiconductor Solutions Corporation*: Sony ofrece una amplia gama de sensores de imagen para diversos segmentos industriales. https://www.sony-semicon.com/en/products/is/industry/index.html
- *Uctronics*: Desarrolla cámaras, lentes y accesorios electrónicos relacionados con Arduino y Raspberry Pi. https://www.uctronics.com
- *OmniVision Technologies*: Ofrece soluciones de imágenes digitales, analógicas y de visualización para múltiples aplicaciones e industrias. https://www.ovt.com
- *Teledyne e2v*: Diseña y fabrica sensores CMOS y su electrónica para soluciones innovadoras en aplicaciones médicas, científicas, aeroespaciales e industriales. https://imaging.teledyne-e2v.com
- *Canon CMOS Sensors*: Canon ofrece sensores CMOS para aplicaciones de visión artificial para industrias, aplicaciones médicas y científicas avanzadas. https://canon-cmos-sensors.com

Para este trabajo de tesis se utilizaron las dos primeras cámaras mostradas en la tabla de la Fig. 2.6, siendo cámaras comerciales sin mayores prestaciones. Las características de estas dos cámaras superan los requerimientos para este proyecto de tesis, ya que se trabajará con imágenes de 640×480 píxeles con una tasa de fotogramas no mayor a 15 fps. Además, no se requiere de un rango dinámico o sensibilidad particular ya que el nivel de iluminación en donde se utilizará se encuentra controlado, ya sea en un ambiente laboral de oficina, en laboratorio o en planta industrial con las condiciones adecuadas para el trabajo humano sin que la iluminación afecte el desempeño de las personas en sus tareas.

2.2. Técnicas

En esta sección se introducen algunas definiciones y se explican distintas técnicas que permiten describir los medios utilizados para realizar la detección de las manos, identificar la postura y con ello diseñar un sistema de mando para la interacción humano-computadora utilizando cámaras RGB.

2.2.1. Aprendizaje Automático

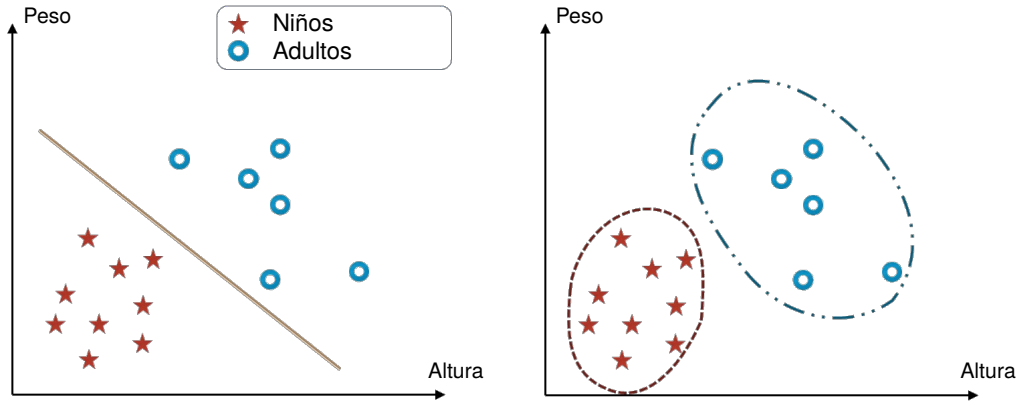
El aprendizaje automático (ML: Machine Learning) se focaliza en el desarrollo de programas informáticos que pueden cambiar cuando se exponen a nuevos datos. Se utilizan algoritmos para identificar patrones y características en los datos para crear modelos[12] que permitan realizar predicciones. Con mayor cantidad de datos de ejemplo, en general, los resultados del aprendizaje automático son más precisos logrando mejores resultados.

Técnicas de aprendizaje automático

Las dos técnicas más importantes en las que se puede dividir el aprendizaje automático son: *Aprendizaje supervisado* y *Aprendizaje no supervisado.*

El *Aprendizaje supervisado* utiliza datos de ejemplo (o de entrenamiento), de los cuales se conoce la respuesta o clase, es decir, los ejemplos están etiquetados. El método consiste en modificar (o aprender) de manera iterativa los parámetros del modelo para que luego permita realizar predicciones con entradas nunca antes conocidas. Un modelo de clasificación pretende identificar la categoría de los datos. En el ejemplo de la Fig. 2.7a se tiene una serie de personas, de las cuales se conocen sus pesos y alturas, y se desea una clasificación en adultos o niños. El modelo aprenderá dónde están esos puntos y creará un clasificador que, para datos de entrada desconocidos, consiga identificar su categoría o clase. Por otro lado, en el *Aprendizaje no supervisado*, los datos de entrenamiento no se encuentran etiquetados, es decir, no se conoce a qué categoría pertenecen. El algoritmo debe determinar patrones y características en los datos por sí mismo y realizar una agrupación (o clustering) de las entradas según características parecidas. En la Fig. 2.7b se muestra un ejemplo que agrupa los datos según características similares.

[12]Los modelos de machine learning son los motores matemáticos, los algoritmos y las distintas técnicas que encuentran patrones y características en los datos para realizar predicciones, estimaciones y agrupaciones de los datos.

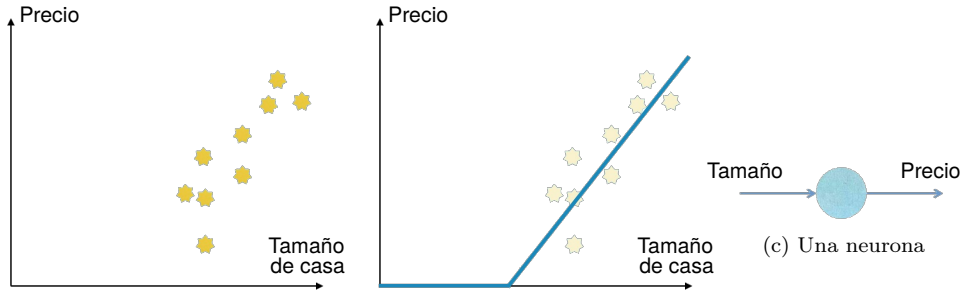


(a) Clasificación (aprendizaje supervisado) (b) Clustering (aprendizaje no supervisado)

Figura 2.7: Aprendizaje automático

2.2.2. Regresión lineal

Es un algoritmo de aprendizaje supervisado que se utiliza en Machine Learning y es apropiado aquí realizar un breve análisis. Se supone que en la Fig. 2.8a se presentan datos sobre el precio de casas según su tamaño. Con estos datos es posible obtener un modelo para luego poder realizar predicciones. La regresión lineal nos entrega dicho modelo (Fig. 2.8b), sabiendo que el precio no puede ser menor que cero, motivo por el cual tiende a cero el precio para casas pequeñas. Esta función puede quedar adaptada a una simple red neuronal como muestra la Fig. 2.8c. El círculo es una neurona que implementa el modelo de regresión lineal. La neurona computa con una función lineal sus entradas (que es el tamaño de las casas), entregando a la salida la predicción del precio.



(a) Datos sobre las casas (b) Función de regresión lineal (c) Una neurona

Figura 2.8: Regresión lineal

La función de regresión lineal muestra que los valores de salida comienzan en cero y luego tienen una pendiente lineal. Esta función es llamada ReLU (Rectified Linear Unit o Unidad Lineal Rectificada) y es una función llamada *función de activación*, las cuales se encargan de devolver una salida a partir de una entrada. Se buscan funciones cuyas derivadas sean simples, para minimizar con ello el coste computacional. Ahora bien, si se desean realizar predicciones más precisas, se tendrían que considerar otras características (features) como la cantidad de habitaciones, ya que se tiene que adaptar a familias con distinta cantidad de integrantes. Entonces, se podría considerar la red de la Fig. 2.9.

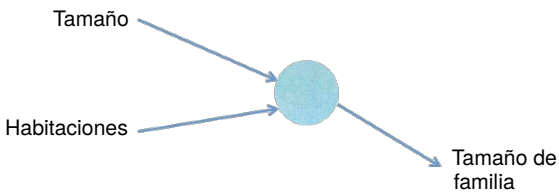


Figura 2.9: Nuevas características para el modelo

También el precio depende de la ubicación (código postal, por ejemplo), ya que lo hace más caminable (walkability), que se encuentre más cercano a los negocios y almacenes hace que no se tenga que utilizar el automóvil, esto también es una característica para predecir el precio. Otra característica puede ser el lujo, por ejemplo, si existen escuelas de mayor calidad en la zona. Entonces, el tamaño y la cantidad de habitaciones predicen el tamaño de la familia, la ubicación predice la caminabilidad y junto con el lujo se predice la calidad de los colegios. Con todo esto (el tamaño de la familia, la caminabilidad y la calidad de los colegios) se puede predecir el precio de la casa. La red quedaría como muestra la Fig. 2.10. Cada una de las neuronas pueden utilizar ReLU u otras funciones de activación. En esta red neuronal hay 4 entradas (variable x) y la predicción es el precio (variable y) y toda la pila de neuronas (que podría ser una sola también) forman la red neuronal.

Ahora se necesitan datos reales para realizar el entrenamiento. La Fig. 2.11 muestra la implementación de la red neuronal, la cual está densamente conectada porque, por ejemplo, la predicción del tamaño de la familia no depende sólo del tamaño de la casa y de la cantidad de habitaciones, sino que también de la ubicación y el lujo. Para realizar las predicciones se debe primero realizar un entrenamiento con suficientes datos de entrada con su correspondiente salida. Aquí está el motivo por el cual es llamado aprendizaje supervisado, ya que se dispone de datos de ejemplo (o de entrenamiento) de los cuales ya se conocen las respuestas. En el caso de las imágenes, se tendrían muchas imágenes en las cuales ya se conocen los objetos que allí se encuentran.

Regresión logística

A diferencia de la regresión lineal, que permite predecir valores continuos, en la regresión logística se puede interpretar la salida como la probabilidad de que una muestra pertenezca a una clase. El método de regresión logística es muy utilizado para resolver

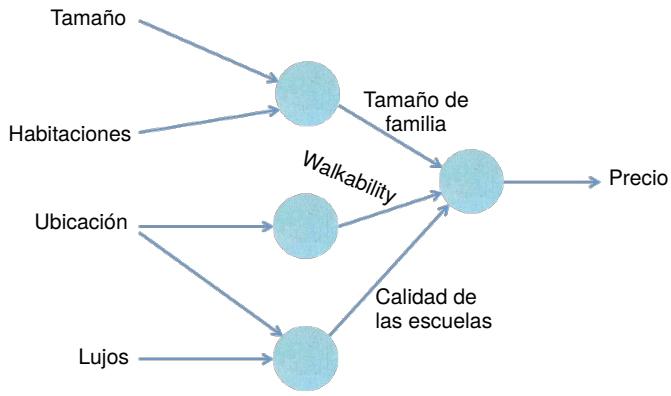


Figura 2.10: Red para predecir precios de casas

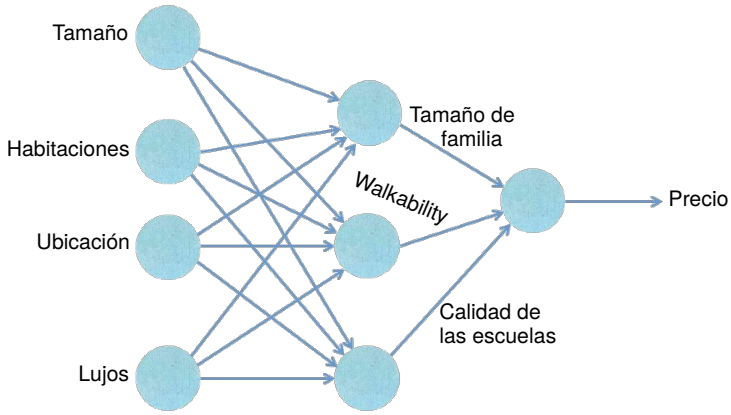


Figura 2.11: Implementación de la red neuronal

problemas de clasificación binaria, donde el resultado sólo puede tomar dos valores posibles. Lo que se muestra en la Fig. 2.12 es un problema de clasificación binaria, donde la salida puede ser "Gato" o "No Gato".

Función de activación

Una función de activación (o función de transferencia) entrega una salida a partir de uno o más valores de entrada. Entre las funciones más utilizadas se encuentran: Función sigmoide, ReLU y softmax.

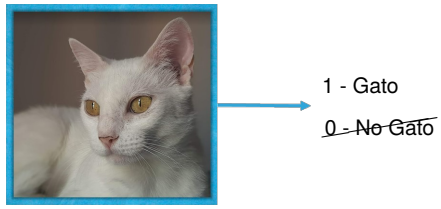


Figura 2.12: Clasificación binaria

Función sigmoide

Se trata de una función de activación que transforma sus valores de entrada de una manera suave y asintótica, proporcionando como resultado un rango de salida comprendido entre (0, 1). Se muestra en la Fig. 2.13.

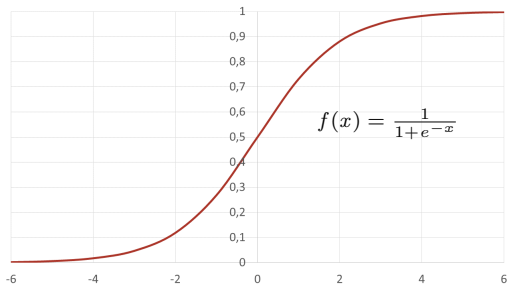


Figura 2.13: Función sigmoide

ReLU

La función ReLU (Rectified Lineal Unit) transforma los valores introducidos anulando los valores negativos y dejando los positivos tal y como entran. Es decir, simplemente hace max(0, x) y su gráfica se muestra en la Fig. 2.14.

Softmax

La función softmax transforma las salidas a una representación en forma de probabilidades, de tal manera que el sumatorio de todas las probabilidades de las salidas sea igual a 1. La función softmax no es más que una generalización de la regresión logística cuando se tiene más de dos clases, por lo que es utilizada para problemas de clasificación multiclase. Softmax es comúnmente utilizada como función de activación en la última capa en clasificadores basados en redes neuronales para representar una distribución de probabilidad definida sobre las diferentes clases.

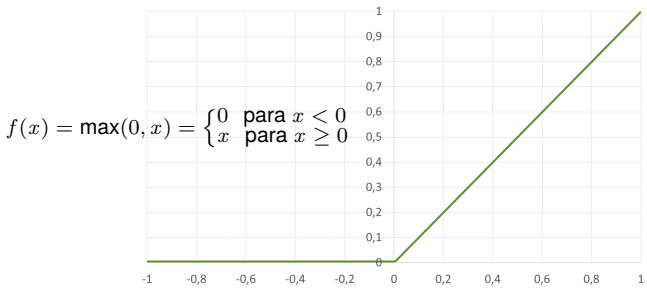


Figura 2.14: Función ReLU

2.2.3. Arquitectura de Redes Neuronales Artificiales

Se denomina arquitectura a la topología o estructura en la que las distintas neuronas constituyentes de la red neuronal se asocian. En general, las neuronas se suelen agrupar en unidades estructurales denominadas capas. Podemos distinguir tres tipos de capas (Fig. 2.15):

- *Capa de entrada*: compuesta por neuronas que reciben datos.
- *Capa oculta*: aquella que no tiene una conexión directa con el entorno.
- *Capa de salida*: neuronas que proporcionan la respuesta de la red.

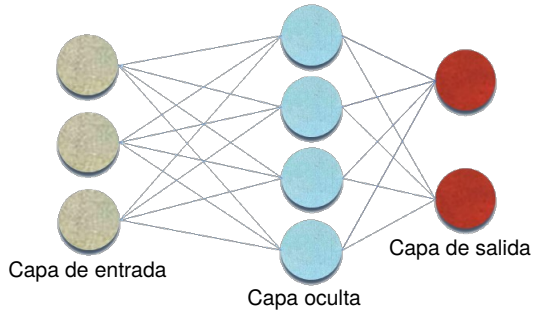


Figura 2.15: Capas de una red neuronal

Vector de características

Un vector de características (feature vector) contiene todos los valores de los píxeles de todos los canales (R, G y B) de una imagen. Por ejemplo, para una imagen RGB de 64×64 píxeles por canal, el feature vector tendría 12 288 elementos, como se visualiza en la Fig. 2.16. La dimensión del feature vector se puede representar con $n_x = 12\,288$ y la notación $x \rightarrow y$ indica que a partir del feature vector x se genera una predicción y.

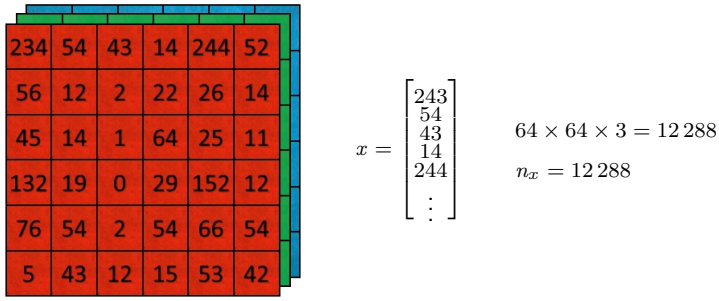


Figura 2.16: Vector de características de una imagen RGB

Si se tienen m imágenes de entrenamiento, los m feature vectors asociados serían $x^{(1)}, x^{(2)}, ..., x^{(m)}$ y sus respectivas predicciones serían $y^{(1)}, y^{(2)}, ..., y^{(m)}$. Los feature vectors y sus predicciones se pueden escribir como sigue:

$$(x^{(1)},\ y^{(1)}),\ (x^{(2)},\ y^{(2)}),\ \dots ,\ (x^{(m)},\ y^{(m)})$$

Se debe tener presente que un feature vector es una imagen, es decir, son todos los valores de cada píxel de cada canal organizados en un vector de, por ejemplo, $64 \times 64 \times 3$ dimensiones.

2.2.4. Redes neuronales profundas

Una red neuronal profunda (DNN: Deep Neural Network) es una red neuronal artificial con dos o más capas ocultas entre las capas de entrada y salida. En la Fig. 2.17 se muestran dos redes neuronales profundas y una que no lo es.

Es muy importante tener en consideración el tamaño de las imágenes que se desean procesar en una red neuronal profunda, ya que el feature vector se puede incrementar notablemente. Como se observa en la Fig. 2.18, una imagen de 64×64 píxeles tiene un feature vector de 12 288 elementos $(x_1,\ x_2,\ \dots ,\ x_{12\,288})$ y una imagen de $1\,000 \times 1\,000$ tiene 3 000 000 de elementos $(x_1,\ x_2,\ \dots ,\ x_{2\,999\,999},\ x_{3\,000\,000})$.

Otras de las variables en juego son las matrices de pesos. Normalmente una neurona recibe múltiples entradas y cada una de estas entradas tiene su propio peso, el cual

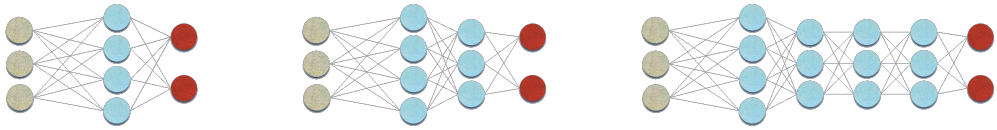


Figura 2.17: Capas ocultas de las redes neuronales

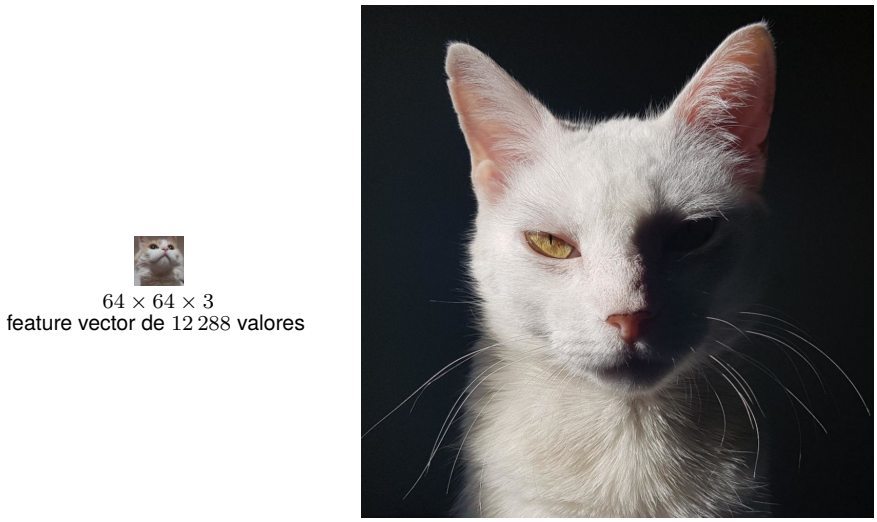


Figura 2.18: Tamaño del feature vector

representa la importancia de dicha entrada en la función de activación de la neurona. Si, por ejemplo, en la primer capa oculta existen 1 000 unidades (1 000 neuronas), entonces el número de pesos en la matriz de pesos usando una red estándar o la llamada Fully Connected Network será de dimensión 3 000 000 × 1 000. Por lo tanto, serán 3 mil millones (3 000 000 000) de parámetros, los cuales requerirán grandes capacidades de procesamiento y de memoria para entrenar una red con esa cantidad de parámetros. Será necesario mejorar esta situación implementando operaciones convolucionales para crear las denominadas Redes Neuronales Convolucionales (CNN: Convolutional Neural Network o ConvNet).

Con las redes neuronales convolucionales pueden realizarse distintas tareas de visión artificial, como clasificación de imágenes, detección y segmentación de objetos (Fig. 2.19).

Figura 2.19: Clasificación, detección y segmentación

2.2.5. Operaciones convolucionales

A continuación se analiza la operación convolucional tomando como ejemplo la detección de bordes verticales y horizontales en una imagen (Fig. 2.20).

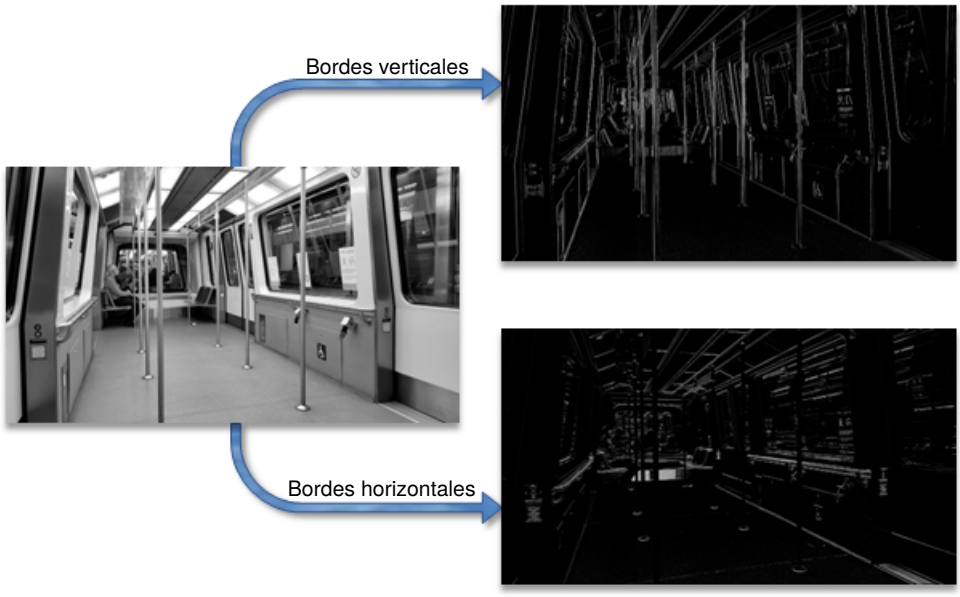


Figura 2.20: Detección de bordes verticales y horizontales

Detección de bordes verticales

Se pone como ejemplo una imagen en escala de grises de 6×6 píxeles, es decir, una matriz de $6 \times 6 \times 1$ y una matriz de 3×3 (llamada filtro, filter o kernel) para lograr la detección (Fig. 2.21).

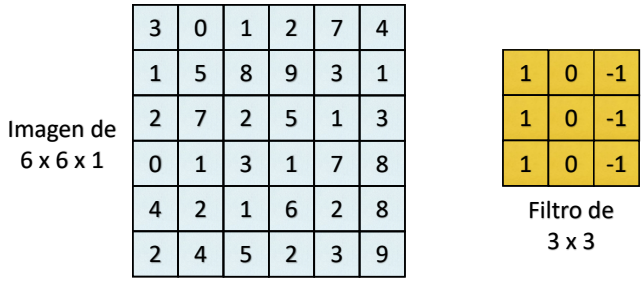


Figura 2.21: Imagen de $6 \times 6 \times 1$ y kernel de 3×3

La convolución se denota con $*$ o con $\otimes$ y la salida en este caso será una matriz de 4×4, en la cual el primer elemento saldrá de la operación mostrada en la Fig. 2.22 y la matriz resultante de las convoluciones de todos los elementos en la Fig. 2.23:

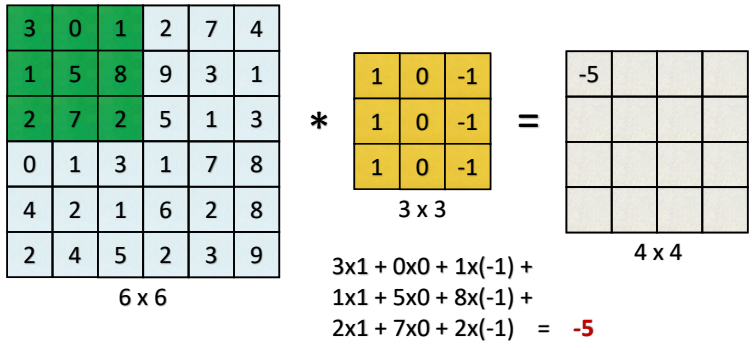


Figura 2.22: Resultado de convolución del primer elemento

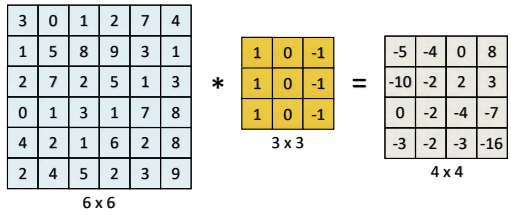


Figura 2.23: Resultado de convolución de todos los elementos

Es necesario tener en cuenta que en los lenguajes de programación y en las bibliotecas usadas para redes neuronales, deep learning y computer vision pueden encontrarse diferencias entre la notación con el asterisco para el operador de convolución y el producto de matrices.

En la Fig. 2.24 se visualizan dos casos de detección de bordes verticales en donde se muestran los valores de las imágenes de entrada, los valores de un filtro para detección de bordes y los valores de la matriz resultante de la convolución entre ambas. Por debajo de estas matrices se observan sus correspondientes representaciones visuales. Notar en ellas que los valores negativos son más oscuros que el valor cero, el cual está representado por el color gris. Observando la representación visual de los resultados de las convoluciones, se puede advertir que el filtro para detección de bordes no sólo entrega información de la posición del borde en la imagen sino también la dirección de la transición desde los píxeles más claros a los más oscuros en los bordes detectados. En el caso 1, los valores 30 indican la detección de un borde y como dichos valores se encuentran entre valores menores (0 en este caso), significa que la transición entre los píxeles más claros y los más oscuros en el borde detectado en la imagen de entrada tiene dirección de izquierda a derecha. En cambio, los valores -30 en el caso 2 indican que la transición tiene una dirección de derecha a izquierda.

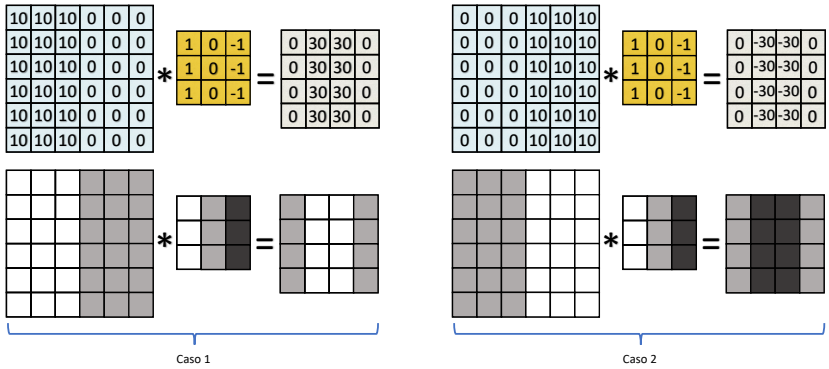


Figura 2.24: Representación visual de los bordes

Existe suficiente literatura sobre los distintos filtros que se pueden usar para detectar los bordes en las imágenes, como el filtro de Sobel, de Scharr, entre otros (Fig. 2.25).

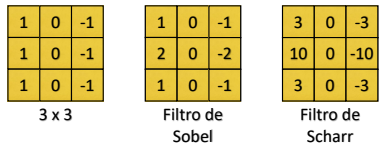


Figura 2.25: Ejemplos de filtros para detectar bordes

Cuando las imágenes para procesar son más complejas (muchos detalles o gran variedad de valores en sus píxeles), el problema está en la selección más cuidadosa de los filtros para detectar los bordes. Para el caso de filtros de 3×3, sus 9 valores pueden ser aprendidos con el método o algoritmo de retropropagación o *Backpropagation*. Este algoritmo juega un papel vital para las redes neuronales, el cual permite entrenar las capas ocultas. Es un método de entrenamiento supervisado que busca ajustar los pesos de manera que disminuya el error entre la salida deseada y la respuesta de la red. La salida deseada es parte de la información que se dispone de los datos de entrenamiento, es decir, se tienen datos de entrada con su correspondiente salida. Este algoritmo de aprendizaje backpropagation conlleva una fase de propagación hacia adelante y otra fase de propagación hacia atrás. Ambas fases se realizan cada vez que se presenta una entrada a la red.

Padding

Al realizar la convolución de una imagen de 6×6 píxeles con un filtro de 3×3, el resultado es una matriz de 4×4, ya que el filtro puede ubicarse en sólo 4×4 posiciones distintas dentro de la imagen. En la Fig. 2.26 se muestran los cálculos que permitirían generalizar las convoluciones.

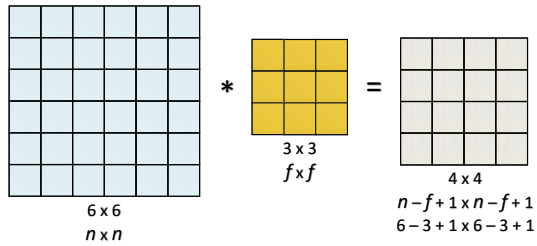


Figura 2.26: Cálculos para la dimensión resultante

Además de reducir el tamaño de la matriz resultante luego de convolucionar, también se puede notar que hay algunos píxeles que intervienen en menor proporción que otros. Por ejemplo, en la Fig. 2.27 se observa que el píxel verde sólo interviene en una única operación, en cambio, el píxel rojo interviene en varias operaciones.

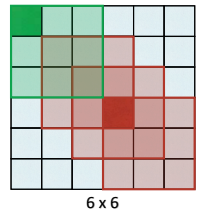


Figura 2.27: Píxeles que intervienen en mayor y menor proporción

Para minimizar estas desventajas que trae la operación de convolución, se hace el *padding* (o relleno), a través del cual se agrega un borde de 1 o más píxeles adicionales (por convención se usan píxeles en cero). Lo que daría como resultado una salida con la misma dimensión que la entrada (ver Fig. 2.28). El píxel verde con padding de 1, influirá en 4 píxeles de salida, que será un poco mayor que sin padding pero igualmente será menor su influencia que el píxel rojo. Si se desea tener mayor influencia, se puede usar un padding mayor.

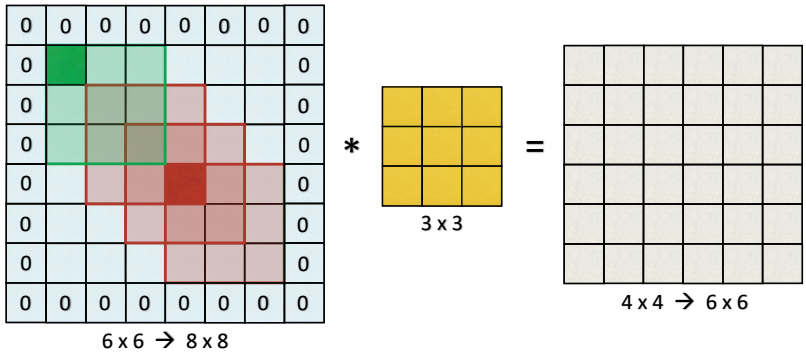


Figura 2.28: Padding

Si el padding es denotado por $p = \text{padding} = 1$, entonces la dimensión de salida será:

$$\begin{array}{lll} n+2p-f+1 & \times & n+2p-f+1 \\ 6+2\ -3+1 & \Rightarrow & 6\times 6 \end{array}$$

Las convoluciones, de acuerdo al padding, se pueden clasificar en:

- *Valid convolution*: Sin padding.

$$\begin{array}{lllll} n\times n & \otimes & n\times n & \Rightarrow & n-f+1\ \times\ n-f+1 \\ 6\times 6 & \otimes & 3\times 3 & \Rightarrow & 4\times 4 \end{array}$$

- *Same convolution*: El padding produce una salida que tiene el mismo tamaño que la entrada.

$$\begin{array}{lll} n+2p-1+f & \times & n+2p-1+f \\ \cancel{n}+2p-1+f & = \cancel{n} \quad \Rightarrow & p=\frac{f-1}{2} \end{array}$$

Stride

El stride (paso) es el desplazamiento (tanto vertical como horizontal) del filtro para realizar la convolución. Por ejemplo, en la Fig. 2.29 se muestra un stride horizontal de valor 2.

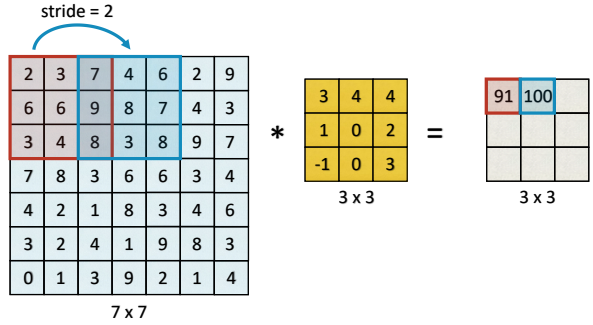


Figura 2.29: Stride

Imagen: $n \times n = 7 \times 7$
Filtro: $f \times f = 3 \times 3$
Padding: $p = 0$
Stride: $s = 2$

$$\frac{n+2p-f}{s}+1 \quad \times \quad \frac{n+2p-f}{s}+1 \qquad \frac{7+0-3}{2}+1=\frac{4}{2}+1=3 \qquad \Rightarrow 3\times 3$$

Cross-Correlation vs Convolution

Es necesario tener presente que, por convención, se llama convolución a la operación que se utiliza en esta temática, pero en realidad esta operación es una correlación. Entonces, por convención en Machine Learning se llama Convolution Operation a la Cross-Correlation Operation. Según la definición matemática de la convolución, se debería hacer lo que muestra la Fig. 2.30.

2.2.6. Convolución en imágenes RGB

En la Fig. 2.31 se muestra una imagen de 3 canales, la cual se convoluciona con un filtro de $3 \times 3 \times 3$. Notar que la imagen resultante es de un sólo canal y tener en cuenta que es frecuente encontrar un cubo como representación gráfica cuando se utilizan varios canales. En este ejemplo, el filtro tiene 27 valores (27 parámetros) que se multiplican canal por canal y se obtiene un único valor con destino a la matriz resultante. Es decir, se hacen las 27 multiplicaciones y luego se suman todas para obtener el valor del primer píxel de la matriz de 4×4.

Si se desea detectar bordes verticales únicamente en un canal, por ejemplo el rojo, entonces el primer canal del filtro deberá ser el filtro de detección de bordes verticales y los demás canales del filtro pueden ser ceros (Fig. 2.32).

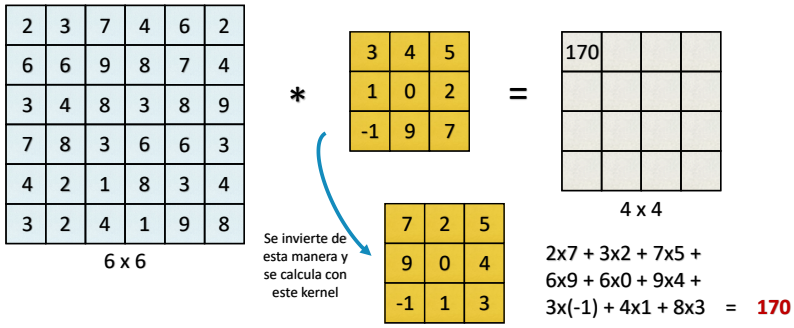


Figura 2.30: Convolución según la definición matemática

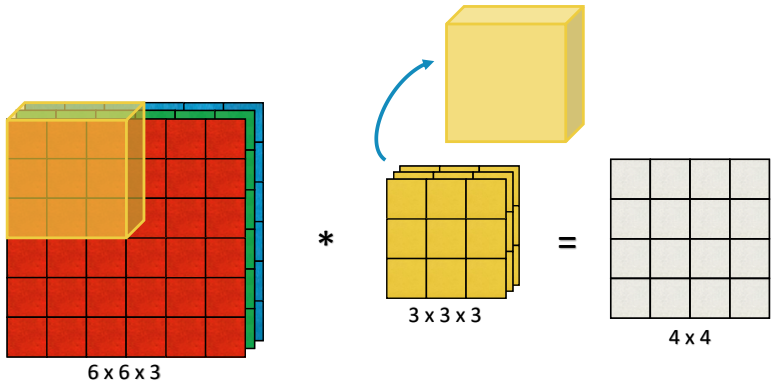


Figura 2.31: Convolución con 3 canales

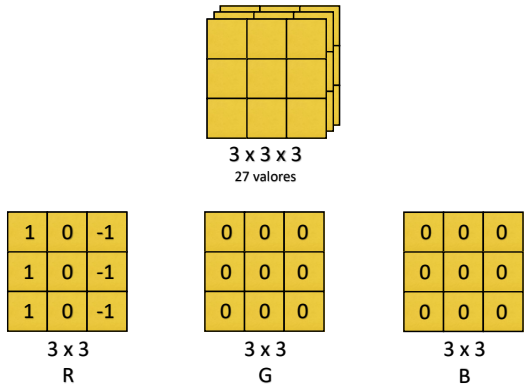


Figura 2.32: Filtro para detección de bordes en canal rojo

También se puede hacer un apilado (stacking) de las salidas, dependiendo de la cantidad de filtros que se estén aplicando. Por ejemplo, se puede necesitar aplicar un detector de bordes verticales y otro de bordes horizontales y se tendrá un volumen de salida de $4 \times 4 \times 2$ como muestra la Fig. 2.33. A esta salida de $4 \times 4 \times 2$ también se la puede dibujar como un cubo. Notar que, en este caso, el número 2 representa a la cantidad de filtros que se aplicaron.

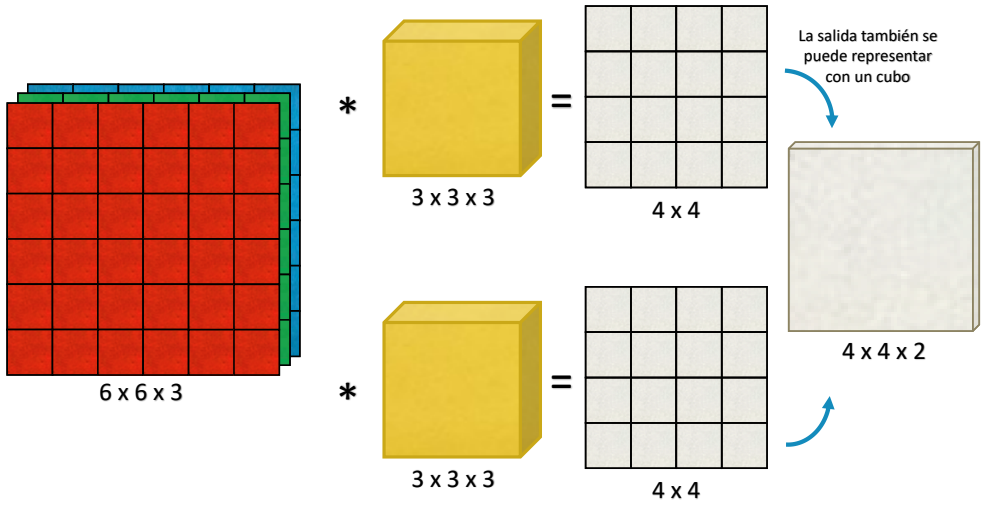


Figura 2.33: Filtros apilados

Respecto a la notación:

n_c = Número de canales
n'_c = Número de canales que serán usados en la convolución siguiente

Se supone aquí que se usa stride = 1 y sin padding.

$$\begin{array}{lllllllll} n \times n \times n_c & \otimes & f \times f \times n_c & \Rightarrow & n-f+1 & \times & n-f+1 & \times & n'_c \\ 6 \times 6 \times 3 & \otimes & 3 \times 3 \times 3 & \Rightarrow & 4 & \times & 4 & \times & 2 \end{array}$$

$n'_c = 2$ = canales para convolución siguiente

Construcción de una capa de una red neuronal convolucional

En la Fig. 2.34 se muestra la convolución entre una imagen de 3 canales con dos filtros de 3 canales cada uno. A cada una de estas salidas 4×4 se le agrega un bias (sesgo), que es un número real y tiene como objetivo mejorar las propiedades de convergencia de la red. Además se le aplica una no linealidad, por ejemplo, la función ReLU. La salida de $4 \times 4 \times 2$ será la entrada para la próxima capa convolucional.

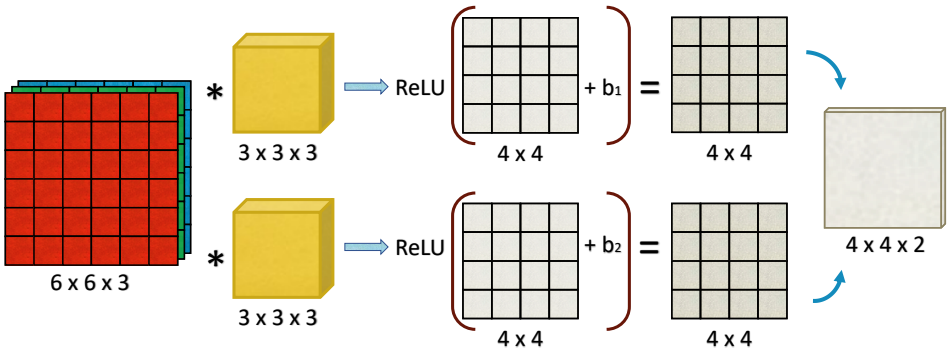


Figura 2.34: Primera capa de la red

Respecto a la notación:

- Con l se denota al número actual de la capa. Se usará como superíndice para indicar a qué capa corresponde.
- $l-1$ es la capa anterior, que sería la activación de la capa l.
- La imagen de entrada a una capa sería la activación de dicha capa.
- El ancho y alto de cada imagen pueden ser distintos, w es el ancho y h es el alto.

l = número de la actual capa de convolución
$f^{[l]}$ = tamaño del filtro $\quad p^{[l]}$ = padding $\quad s^{[l]}$ = stride
$n_c^{[l]}$ = cantidad de filtros de la capa l
$n_c^{[l-1]}$ = cantidad de filtros de la capa $l-1$
Entrada: $n_h^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$
Salida: $n_w^{[l]} = \left[\frac{n_w^{[l-1]} + 2\ p^{[l]} - f^{[l]}}{s^{[l]}} + 1\right] \quad n_h^{[l]} = \left[\frac{n_h^{[l-1]} + 2\ p^{[l]} - f^{[l]}}{s^{[l]}} + 1\right]$
Cada filtro es: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$
Activación: $a^{[l]} \rightarrow n_h^{[l]} \times n_h^{[l]} \times n_c^{[l-1]}$
Pesos: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$
Bias: $n_c^{[l]} - (\ 1,\ 1,\ 1,\ n_c^{[l]}\ )$

2.2.7. Clasificación con Red Neuronal Convolucional

Se muestra un diseño ejemplo de una arquitectura de red neuronal convolucional para la clasificación de imágenes RGB. Pueden existir muchos puntos de partida a la hora de realizar el diseño de una red neuronal, de acuerdo al problema de clasificación que se desee resolver, si se dispone o no de un dataset ya confeccionado para el entrenamiento, la cantidad de clases que se quieren clasificar, el tamaño de las imágenes de entrada y muchas otras cuestiones a considerar. El siguiente ejemplo tiene la finalidad de analizar con cierta profundidad los parámetros que entran en juego.

En la primer capa se tiene una imagen RGB de $39 \times 39 \times 3$ a la cual se le aplican 10 filtros de 3×3 con stride 1 y sin padding. Como muestra la Fig. 2.35, la salida de esta capa es la activación de la próxima capa y tiene una dimensión de $37 \times 37 \times 10$. Notar que la cantidad de filtros define la cantidad de canales de la salida.

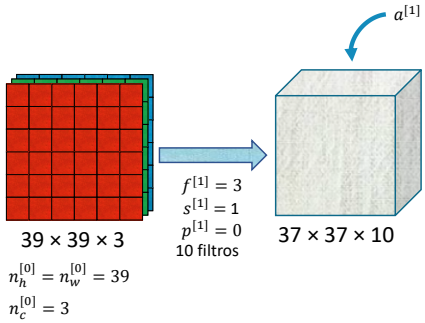


Figura 2.35: Primera capa de la red

La dimensión de la salida se puede calcular con las Ec. 2.1.

$$n_w^{[l]} = \left[\frac{n_w^{[l-1]} + 2\ p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right] \qquad n_h^{[l]} = \left[\frac{n_h^{[l-1]} + 2\ p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right] \tag{2.1}$$

$$n_w^{[1]} = \frac{39+0-3}{1} + 1 = 37 \qquad n_w^{[1]} = \frac{39+0-3}{1} + 1 = 37 \qquad n_c^{[1]} = 10$$

En la segunda capa se usan 20 filtros de tamaño 5×5 con stride $= 2$ y sin padding, obteniendo la activación para la capa siguiente $a^{[2]}$ como muestra la Fig. 2.36.

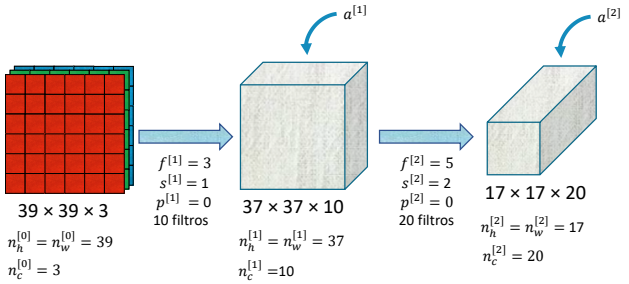


Figura 2.36: Segunda capa de la red

Se puede aplicar ahora una capa con 40 filtros como muestra la Fig. 2.37.

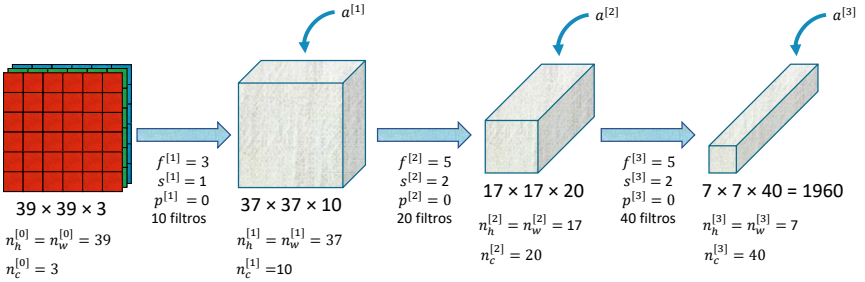


Figura 2.37: Tercera capa de la red

Este último volumen se puede aplanar en 1 960 valores en un vector para luego alimentar con ellos una unidad softmax, y así obtener la salida final que sería la predicción, como muestra la Fig. 2.38.

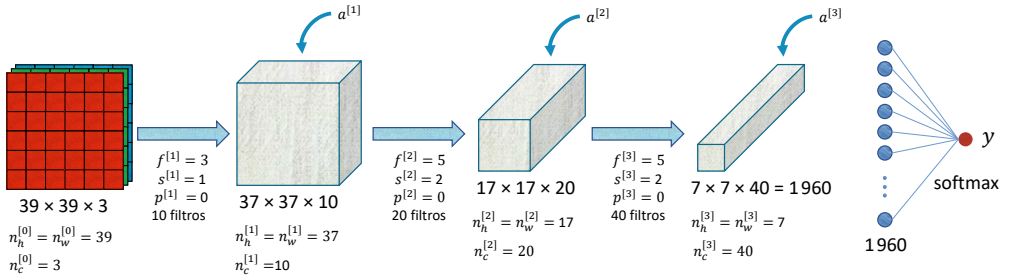


Figura 2.38: Última capa de la red

Hiperparámetros

Cuando se habla de los hiperparámetros de una red neuronal se hace referencia a aquellos parámetros que definen la arquitectura de la red y a los algoritmos que la componen, como ser la cantidad de capas ocultas, el tipo de funciones de activación de las diferentes capas, la cantidad de neuronas o filtros, el padding, stride, entre otros. Los hiperparámetros son valores que los determina el experto en Machine Learning y no se encuentran dentro de los parámetros que se aprenden. El ajuste de estos hiperparámetros, también conocido como optimización de hiperparámetros, es fundamental para alcanzar los mejores rendimientos.

Capa de pooling

Con el objetivo de reducir el tamaño de la representación y así aumentar la velocidad de cómputo, se utilizan capas de pooling (agrupación o muestreo), las cuales se encargan de extraer los elementos representativos de un grupo de elementos. Entre los métodos más utilizados se encuentran: *Max Pooling*, que toma el valor máximo de la región analizada (Fig. 2.39), y *Average Pooling*, que toma el valor promedio de los elementos (Fig. 2.40). Notar que en estos ejemplos se utiliza un stride de 2.

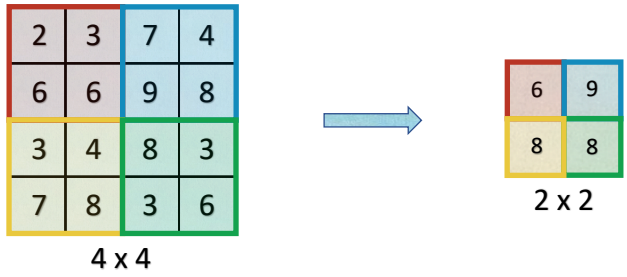


Figura 2.39: Max Pooling

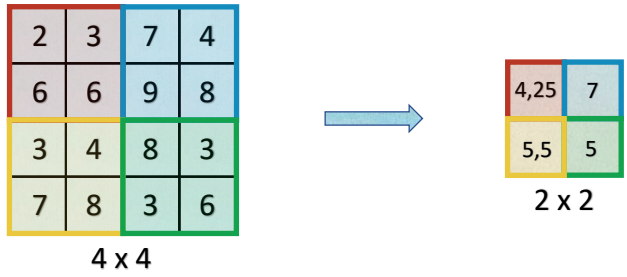


Figura 2.40: Average Pooling

Reconocimiento de dígitos manuscritos

Un ejemplo popular para explicar el diseño de las redes neuronales convolucionales es la arquitectura LeNet-5 [Lecun et al., 1998] creada para la identificación de dígitos manuscritos desde el 0 hasta el 9. Se parte de una imagen de entrada RGB de $32 \times 32 \times 3$ a la cual se le aplican 6 filtros de 5×5 con stride de 1 y sin padding, que se puede observar en la Fig. 2.41. La capa denotada con *Conv 1* corresponde a la aplicación de los 6 filtros de 5×5, la aplicación del bias y la aplicación de una no linealidad, como por ejemplo la función ReLU.

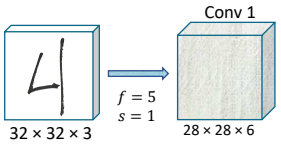


Figura 2.41: Aplicación de 6 filtros de 5×5

Seguido de estas primeras operaciones se puede aplicar una capa de pooling de 2×2, con stride 2, sin padding y utilización de Max Pooling. En la Fig. 2.42 se observa el resultado con la reducción de las dimensiones en un factor de 2 y sin modificar la cantidad de canales. A esta capa de pooling se la puede llamar *Pool 1*. En la literatura, por convención, se agrupan estas dos capas (*Conv 1* y *Pool 1*) en una sola. Todo esto pasa a ser la Capa 1 o Layer 1.

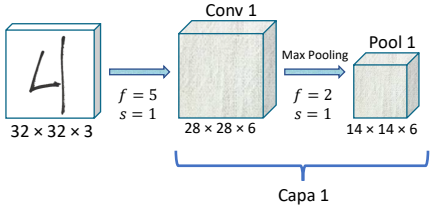


Figura 2.42: Aplicación de Max Pooling

A continuación (Fig. 2.43) se aplican 16 filtros de 5×5, stride $= 1$, sin padding y con Max Pooling. A la salida de la capa 2 se tiene un volumen de $5 \times 5 \times 16$, que son 400 valores, y se pueden aplanar (flatten) a un vector de 400×1.
Luego, se toman esas 400 unidades para construir la próxima capa con 120 unidades (Fig. 2.44). Sería la primera Fully Connected Layer (denotada con FC3) y tiene 400 unidades densamente conectadas a 120 unidades.

Luego, en la Fig. 2.45 se agrega una nueva Fully Connected Layer (FC4) que tiene 84 unidades y finalmente una función softmax, la cual debería ser de 10 salidas, ya que se necesitan reconocer los dígitos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
Tal como lo menciona el autor [Lecun et al., 1998], el experto toma distintas decisiones para confeccionar la arquitectura de la red, guiado por múltiples cuestiones, como ser

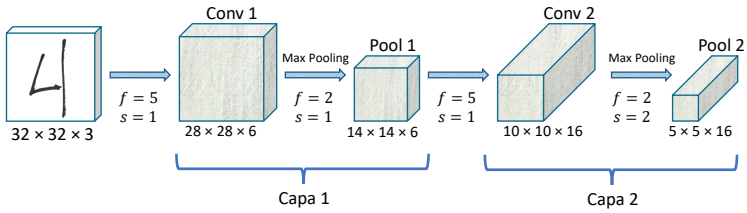


Figura 2.43: Capa 1 y 2

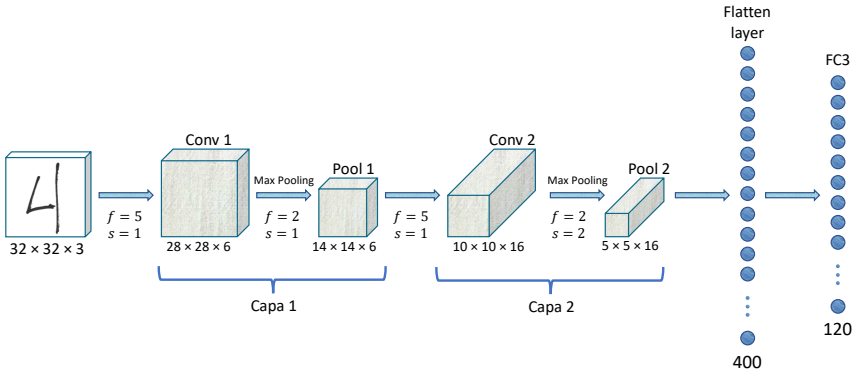


Figura 2.44: Capa densamente conectada

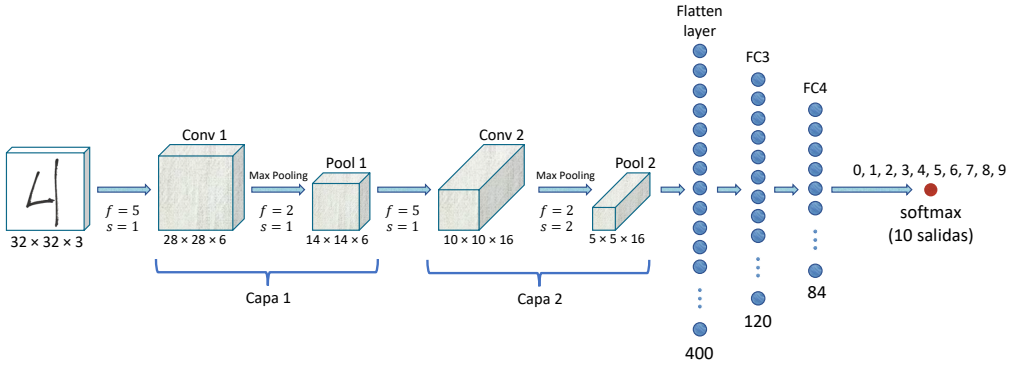


Figura 2.45: Función softmax

el estado del arte, datasets disponibles, capacidad de cómputo, características de las imágenes, entre otras. Para traer alguno de estos motivos se puede mencionar que la capa FC4 fue definida con 84 unidades porque se desea que la red vaya transformando la imagen de entrada en una representación interna de tal forma los dígitos manuscritos queden contenidos en 7×12 píxeles para finalmente pasar a la función softmax.

En la Fig. 2.46 se muestra una tabla con la descripción de la arquitectura de la red como se acostumbra utilizar. Además se colocan aclaraciones sobre la cantidad de parámetros entrenables en cada capa.

	Activation Shape	Activation Size	# parameters
Input	(32, 32, 3)	3072	0
CONV 1 (f=5, s=1)	(28, 28, 8)	6272	208
POOL 1	(14, 14, 8)	1568	0
CONV 2 (f=5, s=1)	(10, 10, 16)	1600	416
POOL 2	(5, 5, 16)	400	0
FC3	(120, 1)	120	48001
FC4	(84, 1)	84	10081
Softmax	(10, 1)	10	841

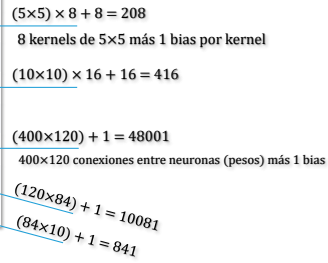


Figura 2.46: Tabla descriptiva de la arquitectura de la red

2.2.8. Perceptrón

La primera publicación [McCulloch and Pitts, 1943] de un modelo computacional de red neuronal fue realizada por William McCulloch y Walter Pitts en 1943 y carecía de la principal funcionalidad que se desea de una red neuronal artificial en la actualidad: *la capacidad de aprender*. McCulloch y Pitts estudiaron el cerebro desde un punto de vista computacional diciendo que una manera de resolver problemas era utilizando una red de unidades interconectadas que, inspirados en la naturaleza del cerebro humano, llamaron neuronas. Estas primeras ideas de redes neuronales permitirían posteriormente que los parámetros de interconexión se ajusten cuando no se obtenga a la salida el valor deseado y, en base a prueba y error, se vayan ajustando hasta obtenerla.

En el año 1957, Frank Rosenblatt inicia su estudio sobre redes neuronales y describe el primer modelo de red neuronal artificial con la capacidad de aprender [Rosenblatt, 1957], y lo llama *perceptrón*, nombre inspirado por su interés en la percepción humana. Rosenblatt diferenciaba entre perceptrones simples con dos capas (una de entrada y otra de salida) y perceptrones multicapa con tres o más capas. También utilizó el término *corrección del error mediante propagación hacia atrás* que luego se adoptaría para definir al algoritmo backpropagation usado no sólo para perceptrones sino también para deep learning.

Los perceptrones simples son clasificadores lineales pudiendo decidir si una entrada pertenece a una clase específica. La frontera de decisión, que separa los ejemplos de una clase de los ejemplos de otra, es una línea recta si se consideran dos dimensiones, un plano para tres dimensiones o, generalizando, corresponde a un hiperplano para n dimensiones.

La frontera de decisión de un clasificador lineal, como lo es un perceptrón simple, es de la forma:

$$b + \sum_i x_i \ w_i = 0$$

Por simplicidad, se puede considerar al bias (sesgo) como un peso más dentro del vector de pesos y que esté asociado a una entrada fija $x_0 = 1$. De esta manera, queda:

$$\sum_i x_i \ w_i = 0$$

siendo:

$$\begin{aligned} &\text{vector de entrada} = (\ 1,\ x_1,\ x_2,\ \dots,\ x_m\ ) && \text{con dimensión } m+1 \\ &\text{vector de pesos} = (\ b,\ w_1,\ w_2,\ \dots,\ w_m\ ) && \text{con dimensión } m+1 \end{aligned}$$

Este tipo de clasificador puede separar clases linealmente separables (ver Fig. 2.47). Se ingresan los datos de entrada en un feature vector para luego asociarlos con los pesos y así obtener un escalar $z = \sum_i x_i \ w_i$ que clasifica al feature vector x. En caso que el valor de z supere el umbral de activación de la neurona, entonces la entrada x corresponde a la clase positiva o clase deseada, siendo $y = 1$, de lo contrario será de la clase negativa, $y = 0$. Para realizar el ajuste de los pesos se procesan uno por uno los ejemplos de entrenamiento del dataset y se actualizan los pesos de la siguiente manera:

- Si la salida es cero y esto es incorrecto (falso negativo), entonces se suma el vector de entrada al vector de pesos.
- Si la salida es uno y esto es incorrecto (falso positivo), entonces se resta el vector de entrada al vector de pesos.
- Si la salida es correcta, los pesos no se modifican.

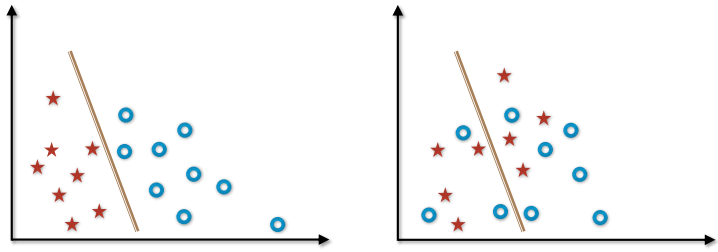


Figura 2.47: Clases linealmente separables y no linealmente separables

De esta manera se recorre varias veces (iteraciones) con los ejemplos del dataset hasta que el algoritmo converja hacia un valor umbral establecido o hasta que no haya mejoras significativas. Durante el entrenamiento se denomina época (epoch) a cada recorrido que se hace pasando por todos los ejemplos del dataset.

Tasa de aprendizaje

La tasa de aprendizaje (o learning rate) es un hiperparámetro para controlar la magnitud de las actualizaciones que se realizan sobre el modelo que se está entrenando. Se suele denotar con η (eta) y sirve para regular la velocidad de actualización de los parámetros o pesos de la red. Cuando se utiliza una función de error (llamada también función de coste o función de pérdida), el learning rate va multiplicado al valor devuelto por la función de error, el cual indica en qué sentido corregir los valores que se desean modificar durante el entrenamiento.

2.2.9. Perceptrón multicapa

Una red neuronal de una sola capa, como el caso del perceptrón, puede realizar clasificación con clases linealmente separables y esto es una limitación. La primera opción es ampliar la cantidad de capas en el diseño de la red, transformándola en una red multicapa (con capa de entrada, una o más capas ocultas y la capa de salida). Si todas las capas ocultas emplearan funciones de activación lineales, entonces todas las capas ocultas se podrían combinar en una única capa, que también sería lineal. Es decir, para alcanzar el correcto desempeño con clases no linealmente separables, necesariamente se deben utilizar al menos algunas funciones de activación no lineales.

Las redes multicapa que tienen la característica de no poseer realimentación se denominan Feed-Forward Neural Networks (FFNN) o red neuronal prealimentada, y también Perceptrón Multicapa (MLP: MultiLayer Perceptron). Es común que las capas estén densamente conectadas (Fully Connected Layers), es decir, que todas las salidas de una capa se conectan a todas las neuronas de la capa siguiente. El entrenamiento de una red del tipo perceptrón multicapa se puede realizar utilizando el algoritmo de backpropagation que permite calcular el gradiente del error con respecto a los diferentes parámetros de la red, es decir, permite identificar de qué manera varía el error conforme se ajustan los parámetros de la red. El gradiente se calcula para una función de pérdida, y un método de optimización, como el gradiente descendente, ajusta los pesos de la red con el fin de minimizar esa función de pérdida.

Para la definición y elección del diseño de una red multicapa se deben tener en cuenta muchos aspectos, como ser: arquitectura de la red, cantidad de capas ocultas, neuronas por capa, cómo se ajustan los pesos de la red, learning rate, épocas, cómo se inicializan los pesos, entre tantas otras características. Estos parámetros se los denominan hiperparámetros, con el objetivo de distinguirlos de los parámetros de la red, aquellos que se ajustan durante el entrenamiento.

2.2.10. MediaPipe

MediaPipe es un framework[13] para construir pipelines[14] para realizar inferencia sobre datos (audio y video) con la utilización de algoritmos y modelos de machine learning. MediaPipe facilita la construcción de prototipos para realizar las pruebas necesarias hasta poder convertirlos en aplicaciones finales. MediaPipe es parte de Google Research[15], sigue el concepto de colaboración abierta (open source) y es multiplataforma.

Algunas de las soluciones de visión artificial que se pueden encontrar en MediaPipe[16] son:

- **Hands**: Detección de manos y detalles de articulaciones y falanges (Fig. 2.48).

Figura 2.48: Mediapipe - Hands

- **Face Mesh**: Estimación de una malla que envuelve el rostro (Fig. 2.49).

Figura 2.49: Mediapipe - Face mesh

[13]Un framework o entorno de trabajo está formado por herramientas para el desarrollo de software, incluyendo conceptos particulares, bibliotecas de programación, algoritmos y técnicas para el desarrollo de soluciones específicas.
[14]Un pipeline consiste en una secuencia de procesos informáticos conectados, que pueden ser: comandos, ejecuciones de programas, tareas, subprocesos, procedimientos, entre otros.
[15]Google Research - `https://research.google`
[16]MediaPipe - https://google.github.io/mediapipe

- **Face Detection**: Detección del rostro (Fig. 2.50).

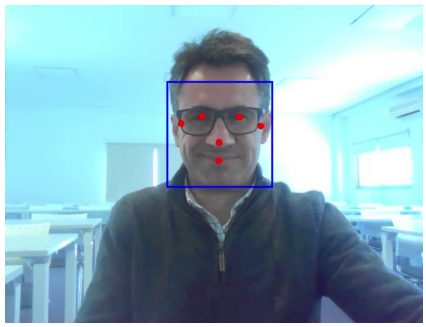


Figura 2.50: Mediapipe - Face detection

- **Pose**: Detección del cuerpo humano y detalles de las articulaciones y huesos principales del esqueleto (Fig. 2.51).

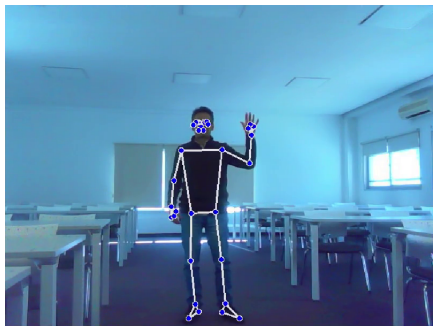


Figura 2.51: Mediapipe - Pose

Respecto a la solución de MediaPipe [Zhang et al., 2020] referida a la detección de las manos, es una propuesta atractiva para el seguimiento de las manos con cámaras RGB en tiempo real[17]. Esta solución realiza predicciones de ubicación del esqueleto de las manos dentro de imágenes RGB. Está desarrollado en base a dos modelos: un detector de la palma de la mano entregando un bounding box[18] donde se encuentra la mano, y un modelo que extrae 21 puntos de referencia (o landmarks o keypoints) del esqueleto de la mano.

[17]Cuándo un sistema actúa en tiempo real y cuándo no, es un tema controvertido y las definiciones de tiempo real pueden ser contradictorias y confusas. "Las cosas tienen que hacerse a tiempo, si no, no tienes un sistema de tiempo real". Se considera en esta tesis que un sistema trabaja en tiempo real cuando se garantiza que se ha terminado de procesar cada imagen antes de una latencia previamente establecida. Esta latencia máxima se fija en 125 milisegundos (8 fps).

[18]Bounding box o caja delimitadora es un término muy utilizado para referirse, en la detección de objetos, a un rectángulo en donde se encierra al objeto detectado.

MediaPipe Hands puede realizar el seguimiento de múltiples manos y lo logra con los dos modelos de machine learning trabajando juntos:

- Detector de la palma: Este modelo trabaja con la imagen RGB de entrada completa, identifica la palma y devuelve el bounding box. El modelo está entrenado con imágenes de palmas de las manos en lugar de la mano completa para evitar complejidades ocasionadas por la alta cantidad de posturas distintas que pueden tomar los dedos.

- Detección de 21 keypoints: Este modelo trabaja únicamente sobre la región de los bounding boxes y devuelve los 21 keypoints en 2.5D (dos dimensiones y media[19]).

Una de las estrategias cuando se trabaja en tiempo real es que el modelo detector de la palma sólo se aplica a la primera imagen (frame o fotograma) de toda la secuencia de fotogramas, de esta manera se reduce el tiempo de procesamiento, ya que luego de ubicar la palma sólo se utilizará el modelo de detección de keypoints, pasando como parámetro el punto central del bounding box del fotograma anterior. Cuando se determine que la palma ya no se encuentra en escena, se utilizará nuevamente en modelo detector de palmas.

El modelo utilizado para la detección de la palma es el llamado Single Shot MultiBox Detector [Liu et al., 2016] (conocido por las siglas SSD). Este es un modelo de red neuronal convolucional que permite detectar múltiples objetos en una imagen mediante un único disparo, mientras que otros modelos requieren dos disparos, uno para generar las posibles regiones donde se ubican los objetos y otro para identificar el objeto. SSD genera múltiples posibles bounding boxes con diferentes tamaños y relación de aspecto para facilitar la detección de objetos a múltiples escalas.

Luego se realiza la detección de los 21 keypoints en las regiones donde una palma fue detectada. Cada uno de los keypoints contienen las coordenadas (x, y) dentro de la imagen que se procesó y además incluye una profundidad relativa, que se puede interpretar como una estimación de la coordenada z. Los 21 keypoints están ubicados en la mano, como se muestra en la Fig. 2.52. El modelo también devuelve un valor que indica la probabilidad de la existencia de una mano en la imagen procesada.

[19]Dimensión dos y medio (2.5D) hace referencia a proyecciones gráficas sobre un plano (2D) con la apariencia de tener tres dimensiones (3D) cuando en realidad no lo son. Sin embargo, está aceptado llamar keypoints en 2D cuando se tengan coordenadas (x, y) y también keypoints en 3D teniendo coordenadas (x, y, z).

Figura 2.52: Ubicación de los 21 keypoints

Dataset de MediaPipe Hands

Para entrenar el modelo, los autores crearon un dataset con las siguientes características:

- 6 000 imágenes tomadas con fondos muy variados y en diferentes condiciones de luz. Estas muestras no tienen demasiada variedad de gestos y posturas con los dedos.
- 10 000 imágenes desde distintos ángulos y gran variedad de gestos y posturas con los dedos. Las muestras fueron tomadas sin variaciones de fondo y a partir de 30 personas.
- 100 000 imágenes con manos artificiales (o sintéticas) generadas con un modelo gráfico 3D que, por su propia naturaleza 3D, ya incorpora las coordenadas (x, y, z). De esta manera se pudieron crear fondos diversos, amplia variedad en los gestos y posturas de las manos, modificar el grosor de los dedos, tamaño de las manos, cambios en la iluminación y proporcionar diferentes texturas y tonos a la piel. La utilización de esta estrategia permite disponer de la información sobre las coordenadas tridimensionales en las muestras. Además se crearon las muestras desde tres cámaras virtuales diferentes, es decir, la misma mano sintética tomada desde tres ángulos distintos.

2.2.11. Detección de objetos a través de nube de puntos

Conociendo las opciones para obtener una nube de puntos de precisión y con alta densidad, una cuestión interesante es la identificación de los objetos a partir de su correspondiente nube de puntos. Existe una propuesta de red neuronal del tipo perceptrón multicapa, denominada PointNet [Qi et al., 2017], en la cual los datos de entrada son nubes de puntos y las tareas que realiza son la clasificación de objetos, segmentación de las partes y análisis semántico de la escena. PointNet toma directamente la nube de puntos y devuelve las etiquetas de clase para toda la entrada. Cada punto (x, y, z) se procesa de forma idéntica e independiente, por lo que no se requiere un orden particular.

PointNet tiene su implementación dentro de la biblioteca Keras[20] y permite realizar un ejemplo de clasificación de los modelos gráficos 3D que se encuentran en el dataset ModelNet10 [Zhirong Wu et al., 2015], el cual contiene 10 clases de objetos (bathtub, bed, chair, desk, dresser, monitor, night stand, sofa, table, toilet). Para experimentar con la red de clasificación PointNet[21] se realiza un muestreo aleatorio para extraer 2 048 puntos de cada uno de los objetos disponibles en ModelNet10, se entrena la red y en la Fig. 2.53 se visualizan alguna de las predicciones realizadas.

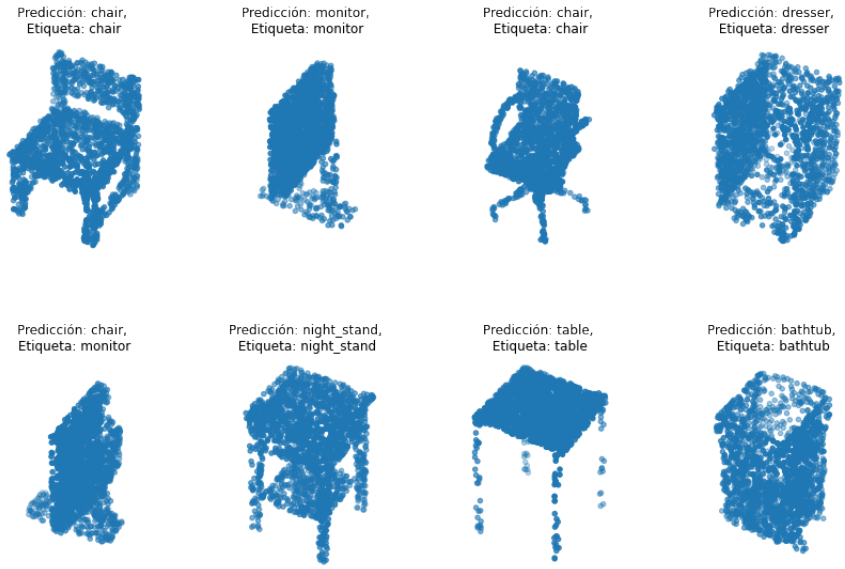


Figura 2.53: Nube de puntos clasificadas

2.3. Estado del arte

El estado último en el tema que aborda esta tesis, llegando a la frontera del conocimiento humano y público, está compuesto por numerosos artículos científicos con importantes avances que año tras año sorprenden con las innovaciones y propuestas. Durante esta sección se realizará una recopilación de los artículos, aislando los temas que tengan relación a esta tesis, como por ejemplo, *Artificial Neural Network*, *Computer Vision*, *Hand Detection*, *Hand Gesture Recognition*, *Point Cloud*, *Keypoints*, *RGB Camera* y *Depth Camera*. Para realizar el análisis del estado de arte se toman en consideración los artículos de los últimos 5 años de las publicaciones científicas de mayor impacto

[20]Keras es una biblioteca de programación escrita en Python para el desarrollo de prototipos con redes neuronales.
[21]Point cloud classification with PointNet - `https://keras.io/examples/vision/pointnet`

dentro de la categoría *Engineering and Computer Science* y sus subcategorías: *Human Computer Interaction* y *Computer Vision and Pattern Recognition*. En el campo de la visión artificial es conocido el prestigio y alto impacto de la *Conference on Computer Vision and Pattern Recognition*. Los autores más reconocidos tienen sus publicaciones allí como también los artículos con mayor cantidad de citas. Con motivo de validar que esto sea así, se recurre a las métricas de *Google Scholar*, que proporcionan un resumen de las citas que tienen las publicaciones científicas, ya sean conferencias, congresos o revistas. Realiza un ordenamiento a través del índice h [22] y permite explorar las publicaciones por áreas de investigación. Para seleccionar las publicaciones científicas de los últimos 5 años, se utiliza el índice h5, el cual hace referencia a los artículos publicados en los últimos 5 años. En las tablas de la Fig. 2.54 se muestran las publicaciones científicas más importantes en el área de investigación que interesan para esta tesis. Se puede notar, en la primer tabla, que la *Conference on Computer Vision and Pattern Recognition* efectivamente es de alto impacto, ya que se encuentra en el Top 5 a nivel general, no sólo en las categorías que interesan para esta tesis.

Todas las áreas de investigación

	Publicación científica	Índice h5
1.	Nature	444
2.	The New England Journal of Medicine	432
3.	Science	401
4.	**IEEE/CVF Conference on Computer Vision and Pattern Recognition**	389
5.	The Lancet	354

Engineering & Computer Science > Computer Vision & Pattern Recognition

	Publicación científica	Índice h5
1.	IEEE/CVF Conference on Computer Vision and Pattern Recognition	389
2.	IEEE/CVF International Conference on Computer Vision	239
3.	European Conference on Computer Vision	186
4.	IEEE Transactions on Pattern Analysis and Machine Intelligence	165
5.	IEEE Transactions on Image Processing	128

Engineering & Computer Science > Human Computer Interaction

	Publicación científica	Índice h5
1.	Conference on Human Factors in Computing Systems (CHI)	113
2.	IEEE Transactions on Affective Computing	62
3.	Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies	58
4.	Proceedings of the ACM on Human-Computer Interaction	57
5.	International Journal of Human-Computer Studies	57

Figura 2.54: Índices h de las publicaciones científicas

La búsqueda de artículos se realizó en torno a las temáticas visión artificial, redes neuronales artificiales e interacción humano computadora, con las siguientes palabras claves: Hand Gesture Recognition, Hand Detection, Point Cloud y Keypoint. Se realizó el análisis de un total de 482 artículos que fueron encontrados en las publicaciones científicas listadas en las tablas de la Fig. 2.54. Se realiza una clasificación de los artículos en relación a los temas que tratan para luego desplegar información de aquellos artículos más relevantes que permitan definir el estado del arte. En la tabla de la Fig. 2.55 se presenta la clasificación realizada y en qué cantidad los artículos se encuentran en las publicaciones científicas seleccionadas de los últimos 5 años.

[22] El índice h de una publicación científica es el mayor número h que hace que al menos h artículos de esa publicación científica sean citados al menos h veces cada uno. Por ejemplo, una publicación científica con 5 artículos citados por, respectivamente, 36, 19, 6, 4 y 2, tiene un índice h de 4.

Clasificación de artículos		
	Temática	**Cantidad**
Manos	Detección y segmentación	30
	Identificación de gestos y posturas	21
	Interacción con objetos	28
Nube de puntos	Clasificación, detección y segmentación	133
	Refinamiento y reconstrucción	82
	Exploración de entorno	29
	Análisis y correspondencia de puntos	79
Keypoints	Keypoints de manos	37
	Cuerpo humano y rostro	13
	Objetos y correspondencia de keypoints	30
	Total	482

Figura 2.55: Clasificación de artículos

A continuación se describen brevemente cada una de estas categorías para mostrar cuáles son las características de los artículos agrupados:

- *Manos / Detección y segmentación:* Artículos para la detección y segmentación de manos. En esta clase de artículos se utilizan redes neuronales convolucionales y otras arquitecturas del estado del arte. Abarca artículos sobre: visión en primera persona; alineación de una mano sintética sobre una mano real; postura de la mano considerando la postura del cuerpo y la ubicación de la cabeza; segmentación de las partes de la mano; segmentación de la mano y el brazo; reconstrucción de la malla de la mano; interacción entre ambas manos; seguimiento de las manos en videos.

- *Manos / Identificación de gestos y posturas:* Agrupa artículos sobre: detección de gestos de la manos durante una conversación; identificación de los gestos en lengua de señas; gestos para el control de interfaces de usuario; detección de gestos en la utilización de un lápiz y el teclado de computadora; dirección hacia donde apunta cada uno de los dedos para dibujar sobre una pantalla; detección de gestos en el uso de un teléfono móvil.

- *Manos / Interacción con objetos:* Artículos sobre: interacción y agarre de objetos; uso de objetos como periférico de entrada; reconstrucción de malla de los objetos agarrados.

- *Nube de puntos / Clasificación, detección y segmentación:* Identificación de los objetos escaneados por LiDAR; segmentación de partes de objetos; identificación de objetos; identificación de bounding boxes de los objetos.

- *Nube de puntos / Refinamiento y reconstrucción:* Aumento de detalles de malla de los objetos; generación de puntos para completar objetos; eliminación de ruido y outliders en nube de puntos; alineación de formas geométricas primitivas.

- *Nube de puntos / Exploración de entorno:* Detección de objetos para conducción vehicular; escaneo de entorno; creación de mapas de entornos explorados.

- *Nube de puntos / Análisis y correspondencia de puntos:* Seguimiento de puntos; correspondencia de puntos; análisis de rotaciones y alineaciones.
- *Keypoints de manos:* Detección de keypoints de las articulaciones de las manos; identificación de la postura de las manos; adquisición de keypoints con imágenes RGB y mapas de profundidad; reconocimiento de acciones dinámicas.
- *Keypoints / Cuerpo humano y rostro:* Detección de keypoints del rostro y cuerpo; ajuste de keypoints; seguimiento de keypoints.
- *Keypoints / Objetos y correspondencia de keypoints:* Detección de keypoints de señales en la vía pública; keypoints de objetos; seguimiento y refinamiento de keypoints.

En primera instancia se realiza una breve revisión de los artículos que tienen menor importancia en relación al tema que interesa en este trabajo, aunque se rescatan algunas propuestas que pueden servir de motivación. Existe una gran cantidad de aportes referidos a nube de puntos y en su gran totalidad utilizan nube de puntos con alta densidad, ya sea con nubes de puntos adquiridas con sensores LiDAR, cámaras de profundidad o directamente desde datasets ya conformados con datos sintéticos o reales. La cantidad de artículos registrados durante esta exploración referidos a nube de puntos es 323. Entre los datos para destacar sobre los trabajos que tratan con nubes de puntos es la cantidad de referencias a la arquitectura de red PointNet, la cual también se utiliza en esta tesis. En general, PointNet es utilizada para realizar comparaciones y validar la arquitectura que se propone en estos artículos. Los datasets más utilizados en esta categoría de artículos son: ModelNet10 y ModelNet40 [Wu et al., 2015], KITTI [Menze and Geiger, 2015], ShapeNet16 [Chang et al., 2015] y DAVIS 2016 [Perazzi et al., 2016]. Es muy común encontrar operaciones de convolución 3D en las arquitecturas de red para trabajar con nube de puntos. También se encuentran algunos pocos artículos que utilizan nubes de puntos de baja densidad, a la que llaman, en lugar de nube de puntos, grupo de puntos, como los artículos [Nam et al., 2021] y [Yang et al., 2019]. En el artículo [Lin et al., 2021] se propone una manera de generar una malla esquelética de objetos con estructuras complejas a partir de su nube de puntos, objetos que también pueden ser manos. En el artículo [Zhao et al., 2022] se plantea una manera de sobremuestrear nubes de puntos para hacerlas más densas y uniformes.

Dejando atrás los artículos relacionados a nubes de puntos, sin haber encontrado algo demasiado específico en relación a nube de puntos de manos, se prosigue con 80 artículos encontrados que involucran keypoints. Algunas propuestas se concentran en obtener keypoints de objetos a partir de imágenes RGB, buscando que esos keypoints representen de la mejor manera la estructura de los objetos. De los artículos sobre keypoints, 37 están enfocados en keypoints de las manos. Existe un grupo de artículos que tienen propuestas similares, en donde crean un modelo de red neuronal entrenado con datasets de imágenes RGB que contienen manos etiquetadas. En la escena de la imagen aparece una mano real o una mano sintética, y el punto de observación, es decir, la posición de la cámara se encuentra en más de una posición. Esta cámara puede ser una cámara RGB real, o

bien, una cámara virtual, es decir, que existe sólo dentro de un entorno de graficación 3D, la cual permite adquirir imágenes multivista sin depender de un laboratorio con cámaras reales instaladas. Las manos de los datasets están etiquetadas (o anotadas) con coordenadas 3D en cada una de las articulaciones de la mano. Si bien, son propuestas similares, se diferencian en la arquitectura de red que plantean. Otros artículos, además de extraer los keypoints, también adaptan y alinean por sobre la mano real, un modelo de mano 3D compuesto por una malla deformable y articulada. Algunos trabajos también recogen el color de la piel para pintar el modelo de mano 3D con el mismo color que la mano real. El modelo 3D de la mano que se encuentra frecuentemente en estos trabajos es el llamado MANO (hand Model with Articulated and Non-rigid defOrmations) [Romero et al., 2017]. Esto antes descrito corresponde a los artículos [Corona et al., 2022], [Chen et al., 2021], [Kulon et al., 2020], [Ge et al., 2019], [Mueller et al., 2018] y [Iqbal et al., 2018]. Es importante destacar que estos trabajo anteriores son similares a MediaPipe Hands [Zhang et al., 2020], que realiza la estimación de los keypoints de las manos a partir de imágenes adquiridas con cámaras RGB. Adicionalmente en esta línea de trabajos que estiman los keypoints de las manos, pero utilizando mapas de profundidad, es decir, con el uso de cámaras de profundidad en lugar de cámaras RGB, y trabajando con redes neuronales convolucionales, predominantemente capas convolucionales 3D, se encuentran trabajos como [Ren et al., 2022], [Malik et al., 2020], [Du et al., 2019], [Chen et al., 2019], [Chang et al., 2018], [Yuan et al., 2018], [Ge et al., 2018] y [Ye and Kim, 2018].

El artículo [Nguyen et al., 2019] propone tener en cuenta la correlación o conexión que tienen las articulaciones adyacentes de la mano y considera que esta información es crucial para el reconocimiento de los gestos. A partir de esta observación, los autores modelaron el esqueleto de la mano en forma de cuadrícula 2D definiendo conexiones de manera diferente a las utilizadas normalmente en otros trabajos. Con esta nueva estructura es posible utilizar capas convolucionales. En la Fig. 2.56 se pueden observar estas conexiones junto a los pesos asociados a los keypoints de la vecindad del keypoint 12. En la capa de salida se utiliza una función softmax que entrega el gesto identificado. En los experimentos se utiliza un dataset llamado Dynamic Hand Gesture [De Smedt et al., 2016] que contiene 14 gestos de las manos con datos obtenidos a través de cámaras de profundidad.

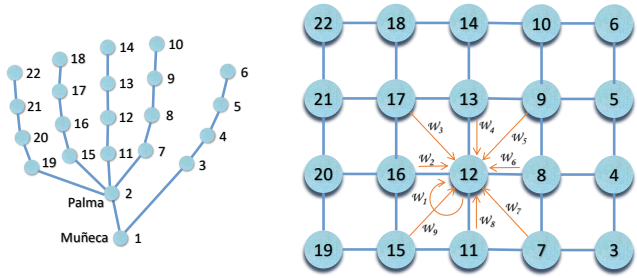


Figura 2.56: Esqueleto de la mano como cuadrícula

Otro tipo de trabajos identifican ambas manos de una persona interactuando entre ellas, como por ejemplo, entrecruzando los dedos o golpeando las palmas. En la propuesta [Kim et al., 2021] se utilizan imágenes RGB y una red neuronal convolucional para detectar los bounding boxes de la mano izquierda, de la mano derecha y de ambas manos interactuando, como se muestra en la Fig. 2.57. Luego se realiza la estimación de la postura de las manos, obteniendo los keypoints gracias a modelos entrenados con mapas de profundidad de manos etiquetadas. Además, se explota la dependencia estructural de las dos manos entrenando un discriminador del tipo GAN[23] (Generative Adversarial Networks) que ayuda a evitar configuraciones de las manos que sean inverosímiles.

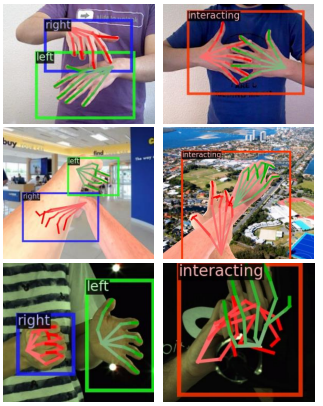


Figura 2.57: Interacción entre manos
Fuente: [Kim et al., 2021]

Entre los datasets que se utilizan en este tipo de trabajos se encuentran YouTube 3D Hands [Kulon et al., 2020], InterHand2.6M [Moon et al., 2020], FreiHAND [Zimmermann et al., 2019], 11k Hands [Afifi, 2019] y EgoHands [Bambach et al., 2015].

En esta exploración se encuentran suficientes artículos que analizan la interacción de las manos con objetos. La mayoría de este tipo de trabajos buscan reconstruir no sólo la malla 3D de las manos, sino también la malla de los objetos con los cuales se interactúa. Esta interacción puede ser agarrando, sosteniendo o tocando los objetos. El desafío es estimar con precisión aquellas superficies de contacto entre manos y objetos, perfeccionando y refinando el correspondiente mapa de contacto obtenido durante la estimación. En general, no se definen nuevos métodos para la estimación de la postura de las manos ni la detección de los objetos, sino que utilizan otros métodos del estado del arte, y se encargan de mejorar los detalles de la interacción entre manos y objetos.

[23]Las GAN son un tipo especial de redes neuronales que generan datos a partir de un conjunto de datos existentes. Su objetivo es analizar y comprender la distribución de los datos de entrenamiento y crear nuevos datos que la sigan de forma fiable. Una GAN está formada por un Generador, que crea los nuevos datos, y un Discriminador que se encarga de calcular la probabilidad de que un dato sea real (del conjunto de entrenamiento), y no sintético (creado por el Generador).

También se encuentran trabajos que identifican una mano sosteniendo un objeto, sólo para hacer foco en el objeto y lograr identificar qué tipo de objeto se está sosteniendo. En el trabajo [Zhou et al., 2020] se presenta un idea interesante, en la cual se identifican distintos objetos que son sostenidos con la mano, y el propio objeto se convierte en una superficie de entrada de comandos usando gestos de los dedos sobre dicho objeto. Es decir, el objeto pasa a ser un periférico de entrada. Otro proyecto que llama la atención es [Pei et al., 2022], en el cual las manos se convierten en objetos virtuales, imitando a los objetos. Por ejemplo, la mano con el pulgar hacia arriba imita a un joystick o la mano con dedo índice y mayor extendidos imita a una tijera. Los autores crearon los diseños de interacción como se observa en la Fig. 2.58.

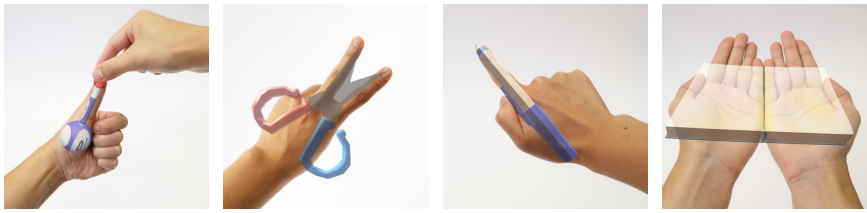


Figura 2.58: Mano imitando objetos
Fuente: [Pei et al., 2022]

Algunos de los datasets más utilizados en estos trabajos están formados por un conjunto de imágenes de manos agarrando objetos, con variaciones en la pose, el fondo, la textura y la iluminación. Estos datasets contienen las imágenes, junto con las mallas de la mano, el modelo 3D del objeto, los keypoints de las manos y mapas de profundidad. Algunos de estos datasets son: DexYCB [Chao et al., 2021], Extended GTEA Gaze+ [Li et al., 2021], YCB-Affordance [Corona et al., 2020], GRAB (GRasping Actions with Bodies) [Taheri et al., 2020] y ObMan (Object Manipulation) [Hasson et al., 2019].

El artículo [Liu et al., 2022b] trata una característica poco estudiada, que es predecir futuras interacciones de las manos y objetos, es decir, predice la trayectoria de la mano y los futuros puntos de contacto con un objeto. El modelo utiliza una red neuronal convolucional para detectar las manos y los objetos en un contexto conocido, ya que utiliza el dataset Epic-Kitchens [Damen et al., 2022] que contiene secuencias de imágenes en primera persona de las actividades diarias en una cocina. De las manos y objetos detectados por la red convolucional se extraen características y pasan a ser la entrada a un Transformer[24] que finalmente entregará el resultado de la predicción. El modelo

[24]Los Transformers [Vaswani et al., 2017] son modelos de aprendizaje profundo, una arquitectura de red neuronal, diseñados para manipular datos secuenciales. Si se consideran las redes neuronales recurrentes, estas emplean un vector ordenado que contiene la información de la secuencia. En cada iteración, a medida que se va recorriendo el vector, el modelo recolecta datos y los va empleando para hacer las predicciones. A medida que la red se hace más profunda, al modelo se le dificulta detectar relaciones entre los datos en etapas tempranas de la secuencia. La arquitectura de Transformers aplica un mecanismo de atención, otorgando un peso a estos datos, permitiendo definir la relación que tienen los datos de la secuencia. Los Transformers utilizan este mecanismo de atención, no sólo para relacionar todos los datos

utiliza una secuencia de imágenes consecutivas de una acción que realiza la persona en su interacción con objetos de una cocina y entrega una predicción de futuras trayectorias de las manos y un mapa de calor del próximo punto de contacto con los objetos.

Ahora queda presentar los últimos 21 artículos, que son los relacionados a la identificación de gestos, es decir, no sólo detectar las manos, segmentarlas, estimar su postura o alinear una malla 3D, sino que, además, identificar un gesto o un movimiento con el cual se quiera realizar alguna acción. De estos artículos podemos mencionar rápidamente aquellos que involucran innovación en su diseño y aplicación, pero que no desarrollan nuevas técnicas para la identificación de los gestos. Es decir, propuestas que utilizan técnicas y herramientas de otros autores para combinarlas y proponer innovación en su aplicación. El trabajo [Kim et al., 2018] utiliza un sensor Leap Motion Controller con su SDK para realizar el seguimiento de la mano y permitir la creación de bocetos 3D con movimientos de las manos en el aire. Luego se utiliza una tableta gráfica Wacom Cintiq 21UX para añadir detalles en ese boceto 3D. En aplicaciones de realidad virtual y realidad aumentada es frecuente tener la necesidad de manipular elementos virtuales tal como ventanas gráficas, imágenes y videos. La gestión de la posición, orientación y escala de estos elementos en un espacio 3D inmersivo puede ser realizado mediante los gestos con las manos. En el trabajo [Lee et al., 2018] se propone una solución mediante gestos dinámicos como por ejemplo, la utilización del dedo índice y el pulgar haciendo un movimiento como una pinza, y con esto agarrar y desplazar las ventanas y demás elementos. Para el desarrollo se utiliza Unity 3D y la implementación se realiza utilizando un casco de realidad virtual Oculus VR para el seguimiento de la cabeza y un sensor Leap Motion Controller colocado en la parte frontal para detectar las manos y además adquirir las imágenes para generar la realidad aumentada. Otras propuestas innovadoras que se pueden encontrar son: colocación de una cámara en el extremo superior de un lápiz para adquirir imágenes de la mano que está escribiendo y poder rescatar algún gesto que se hace con la misma [Matulic et al., 2020]; activación de comandos con las manos colocadas sobre el teclado de una computadora portátil a través de las imágenes adquiridas por una cámara colocada en la parte superior de la pantalla [Chhibber et al., 2021]; pulsera equipada con cámaras infrarrojas que se combinan para detectar los gestos de las manos [Hu et al., 2020]; detección de los gestos de las manos a largas distancias (4 metros aproximadamente) con el uso de cámaras de profundidad [Liu et al., 2022a]; análisis de la coherencia entre los gestos de las manos y la comunicación interpersonal durante conversaciones grupales [Liao and Wang, 2019]; utilización de cámaras de profundidad para la detección de los gestos de las manos cuando están cercanas al rostro como una propuesta para aumentar la capacidad de entrada de datos en el uso de gafas de realidad aumentada [Weng et al., 2021].

Otra rama de investigación es aquella relacionada con la generación de datos, que son muy importantes para la creación de datasets. Con la llegada de las redes GAN (Generative Adversarial Network), muchos estudios se enfocan allí. El artículo [Hu et al.,

de la secuencia, sino también evitar ese diseño recurrente de las redes neuronales recurrentes y poder procesar todas las entradas de la secuencia al mismo tiempo, calculando los pesos de atención entre ellos. Esto permite procesamiento paralelo logrando velocidad y eficiencia durante el entrenamiento.

2021a] utiliza una red GAN para realizar la traducción de un gesto a otro. La información de entrada es una imagen RGB con una mano haciendo un gesto estático. Además de esto, se ingresa un gesto objetivo, pero en este caso no es una imagen sino que es el modelo 3D de una mano, en este caso, el modelo MANO [Romero et al., 2017]. La salida que se desea obtener es otra imagen RGB generada por la red GAN que contenga la misma mano que aparece en la imagen RGB de entrada, pero con el gesto objetivo. Dicho de otra forma, se desea ingresar a la red una imagen con un gesto y que genere una imagen similar en donde el único cambio que exista, sea el gesto de la mano, tal como muestra la Fig. 2.59.

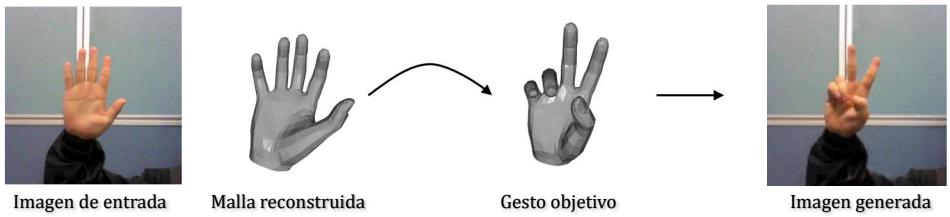


Figura 2.59: Traducción de gesto a otro gesto
Fuente: [Hu et al., 2021a]

Tampoco pueden faltar aquellos artículos que traten la identificación de gestos para lengua de señas [Hu et al., 2021b]. Este trabajo propone un modelo llamado SignBERT, el cual está inspirado en BERT [Devlin et al., 2019] que es un modelo basado en Transformers que, originalmente está pensado para el procesamiento de lenguaje natural (NLP: Natural Language Processing[25]). Para la implementación de SignBERT se utiliza MMPose[26] que permite extraer 23 keypoints 2D de las articulaciones del cuerpo humano, 68 keypoints del rostro y 42 keypoints de las manos (21 por mano). SignBERT, al estar basado en Transformers, utiliza datos secuenciales, por lo que trabaja con una secuencia de imágenes que, luego de usar MMPose, es una secuencia de keypoints. Este modelo entrega la predicción en base al dataset utilizado y el lenguaje de señas utilizado. En este artículo se utilizan los datasets MS-ASL [Vaezi Joze and Koller, 2019] y WLASL [Li et al., 2020] para American Sign Language (ASL), y NMFs-CSL [Hu et al., 2021c] para Chinese Sign Language (CSL).

Finalmente, en esta exploración del estado del arte se mencionan los siguientes artículos que tienen bastante cercanía con esta tesis.

En el artículo [Yang et al., 2020] se propone trabajar de manera colaborativa entre los métodos de detección y compartir información entre ellos para hacer más efectiva la

[25]El procesamiento de lenguaje natural se ocupa de la interacción humano-computadora usando un lenguaje humano. Entre los desafíos se encuentran la comprensión y la generación de lenguaje natural.
[26]MMPose es parte de OpenMMLab, que es un proyecto open source para investigación académica y aplicaciones industriales que cubre tópicos sobre visión artificial y está basado en la biblioteca de aprendizaje automático PyTorch.

detección de, por un lado, el reconocimiento de gestos y por otro, la estimación de la postura en 3D. Aquí es conveniente dejar en claro a lo que se hace referencia con gesto y con postura. Con un gesto se intenta expresar algo, enviar información, comunicar, y una postura es plantarse de alguna manera, adquirir una posición, estar dispuesto de alguna forma. Son muy sutiles estas diferencias, ya que se puede entender que al estar en una postura particular, también se está expresando o comunicando algo. Estas pequeñas diferencias hacen al punto central de este artículo, que se basa en la alta correlación que existe entre un gesto y una postura, es decir, existe mucha dependencia y relación entre gesto y postura. Se puede ejemplificar diciendo que una postura de la mano con el dedo índice totalmente extendido y todos los demás dedos plegados contra la palma, corresponde a un gesto que intenta señalar algo en dirección de la punta del dedo índice. Por este motivo, en este artículo se presenta una nueva red de aprendizaje colaborativo para el reconocimiento de gestos y la estimación de la postura de la mano en 3D, potenciando, mutuamente, estas dos tareas de forma progresiva. El modelo propuesto recibe como entrada imágenes RGB y la salida es el gesto predicho y, además, los keypoints 3D de la mano. Como se observa en la Fig. 2.60, la entrada son las imágenes RGB que ingresan a una red ResNet [He et al., 2016] para aprender los keypoints en 2D de la mano para que ellos sean la entrada de manera simultánea a una subred para las posturas (Pose sub-network) y otra para los gestos (Gesture sub-network). El módulo Pose Feature Analysis está basado en capas convolucionales [Liu et al., 2020] y utiliza los keypoints en 2D de entrada junto con los mapas de profundidad disponibles en el dataset, y con ello logra realizar la estimación de la postura en 3D, es decir, estimar los keypoints en 3D. Los autores utilizan el dataset First-Person Hand Action [Garcia-Hernando et al., 2018], el cual posee 1 175 videos con las manos anotadas con sus keypoints y 45 gestos etiquetados. La salida de la Pose sub-network es la entrada de la Gesture sub-network y operan de forma iterativa. El módulo Gesture Feature Analysis contiene una secuencia de capas convolucionales así como capas de convolución temporal (TCN[27]) para obtener la relación temporal y así predecir el gesto.

Otro interesante trabajo es [Min et al., 2020] en el cual se capturan las correlaciones espaciales en secuencias de nubes de puntos para el reconocimiento de gestos. Con la ayuda de las redes neuronales recurrentes y las LSTM[28] es posible capturar tanto el movimiento como los cambios de apariencia con el paso del tiempo, considerando una mano representada a través de una nube de puntos. Sin embargo, la mayoría de los da-

[27]Una red convolucional temporal (TCN: Temporal Convolutional Networks) hace referencia a las arquitecturas que incorporan capas convolucionales de una sola dimensión. Si bien el modelado de secuencias es sinónimo de redes recurrentes, las redes convolucionales pueden superarlas en estas tareas. En el trabajo [Bai et al., 2018] se realiza una evaluación de las arquitecturas convolucionales y las recurrentes para el modelado de secuencias, y se demuestra un mejor desempeño. Finalmente se presenta una arquitectura genérica que se la bautiza como Temporal Convolutional Networks (TCN).

[28]Las LSTM (Long Short Term Memory) son un tipo especial de redes recurrentes. La característica principal de las redes recurrentes es que la información puede persistir introduciendo bucles en la arquitectura de red, por lo que, básicamente, pueden recordar estados previos y utilizar esta información para decidir cuál será el siguiente. Esta característica las hace muy adecuadas para manejar series cronológicas o secuencias de información. Mientras las redes recurrentes estándar pueden modelar dependencias a corto plazo (es decir, relaciones cercanas en la serie cronológica), las LSTM pueden aprender dependencias largas, por lo que se podría decir que tienen una memoria a más largo plazo.

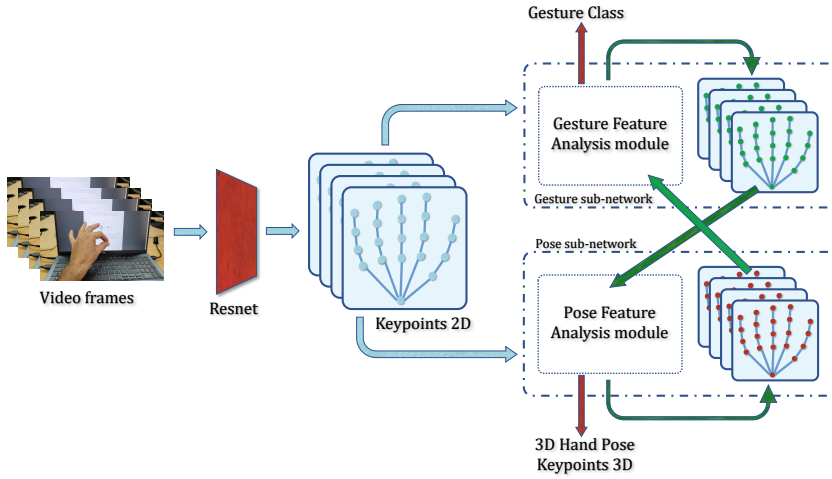


Figura 2.60: Aprendizaje colaborativo para posturas y gestos

tos de las nubes de puntos no tienen orden, y la aplicación directa de LSTM provocaría dificultades. Por lo tanto, para modelar secuencias irregulares preservando la estructura espacial, los autores proponen una red llamada PointLSTM en la cual propagan información del pasado hacia delante conservando la estructura espacial. Para resolver ese problema, para cada punto de la nube en el momento actual corresponde un punto del momento anterior, o en caso de oclusión, para cada punto en el momento actual se puede hacer corresponder una vecindad de puntos en el momento anterior. El número de puntos disponibles en imágenes de profundidad es grande, y la mayoría de ellos contienen información similar. En el trabajo [Min et al., 2019] se muestra que un pequeño número de puntos (entre 100 y 200) es una opción razonable para el reconocimiento de gestos. Por ello y para reducir el cómputo redundante, se realiza un muestreo simple para reducir la cantidad de puntos. El entrenamiento de la red se realiza con una nube de puntos de baja densidad, de 128 puntos, y además se realiza un aumento de datos realizando escalados aleatorios, sin superar un escalado del 20 %, y rotaciones de hasta un 15 %. La propuesta de los autores es el reconocimiento dinámico de gestos, extrayendo secuencias de nubes de puntos de las regiones de las manos, las cuales pueden ser segmentadas gracias a las imágenes de profundidad. Los autores utilizaron los datasets SHREC'17 [De Smedt et al., 2017], NVGesture [Molchanov et al., 2016] y MSR Action3D [Li et al., 2010].

Para cerrar este análisis de artículos, se muestra en la Fig. 2.61 la distribución de los mismos según la clasificación realizada.

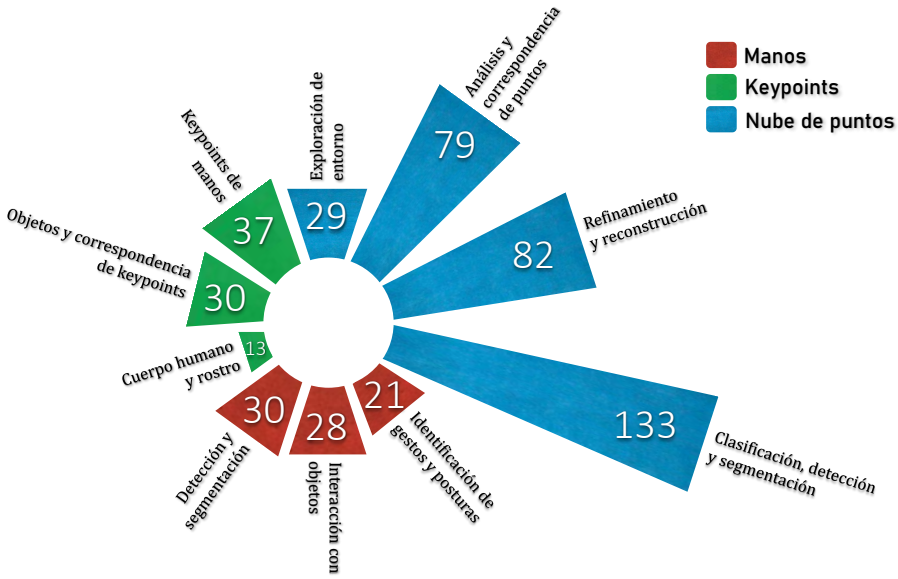


Figura 2.61: Distribución de los artículos clasificados

Luego de esta revisión de artículos[29] de las publicaciones científicas de mayor impacto en los últimos 5 años, es posible mencionar que existen 116 artículos relacionados a las detección de las manos, identificando sus posturas y gestos. Sin embargo, no son demasiados los que atacan el problema de identificar los gestos a partir de las imágenes RGB capturadas por una cámara en tiempo real, realizando las predicciones con una red perceptrón multicapa. Es cierto que existen trabajos muy interesantes y que se aproximan bastante, aunque sin el uso de una arquitectura como la propuesta en esta tesis, la cual es muy simple y de muy pocas capas, alcanzando muy alta tasa de aciertos. De esta manera, se concluye que, la propuesta de esta tesis tiene originalidad y es parte del estado del arte.

[29] El listado completo de los artículos analizados durante esta exploración se encuentra en un anexo online en `https://www.cesarosimani.com.ar/doctorado/anexo_tesis_osimani.pdf`

Capítulo 3

Detección de gestos de las manos

Capítulo 3

Detección de gestos de las manos

Contenidos

En este capítulo se presenta la propuesta de esta tesis, que es una contribución al estado del arte en la temática de detección de gestos de manos con el procesamiento en tiempo real de las imágenes adquiridas por cámaras RGB, para ser aplicada en el diseño de nuevas interfaces de interacción humano-computadora para el control de dispositivos. En el presente capítulo se detalla lo siguiente: obtención de nube de puntos de la mano, elección de los gestos, creación del dataset, normalización de los datos, definición de la arquitectura de la red, entrenamiento y obtención del modelo para las predicciones.

3.1. Nube de puntos de la mano

En base al marco teórico descripto en el capítulo 2 y sabiendo que se dispone de un framework como MediaPipe que entrega una predicción de 21 keypoints en 3D de la mano, sería interesante considerar que estos puntos corresponden a una nube de puntos de la mano. Un atributo importante de las nubes de puntos es la densidad, es decir, la cantidad de puntos disponibles en la nube de puntos. Cuanto mayor sea la densidad de una nube de puntos, más precisa será la descripción de la superficie. Más adelante en este capítulo se analiza la generación de nuevos datos (Data Augmentation), para disponer de una nube de puntos de mayor densidad y así poder analizar el comportamiento de la propuesta de esta tesis con una nube de puntos de baja y alta densidad.

3.2. Selección de gestos

Con el fin de realizar una selección de los posibles gestos que el sistema pudiera detectar, se realiza una exploración de la comunicación no verbal, de lengua de señas, de datasets que incluyan manos y de artículos relacionados a la interacción humano-computadora. En las interfaces naturales de usuario se utilizan diversos gestos con las manos y es habitual encontrar soluciones que utilizan, por ejemplo, la punta del dedo índice o la palma abierta para controlar el mouse; la mano cerrada (puño) seguida de la mano abierta para agarrar y soltar; el pulgar hacia arriba para aceptar o el pulgar hacia abajo para cancelar. Para llegar a definir cuáles pueden ser los gestos que se utilicen en este trabajo, se realiza la caracterización de las manos en base a sus keypoints, con el objetivo de extraer de cada gesto, la posición de sus articulaciones, calculando sus ángulos y, con ello, determinar cuáles son los gestos que más fácilmente se puedan discriminar. Resulta difícil encontrar trabajos que tengan una justificación minuciosa de por qué la elección de los gestos que se utilizan. Aquellos artículos que generan su propio dataset de manos, seleccionan los gestos de manera intuitiva dependiendo el uso que se hará con ellos, y aquellos que utilizan datasets de terceros, simplemente utilizan los gestos que ya vienen en esos datasets. Aquí, se realizará una caracterización de las manos para tener una selección más robusta, con el fin de ayudar a las técnicas de detección que se utilicen en esta tesis, permitiendo disminuir la confusión entre gestos. Para disponer de aquellos gestos que se pueden realizar con las manos sin necesidad de tener una destreza especial,

se obtienen los gestos de los datasets que se muestran en la Fig. 3.1, Fig. 3.2 y Fig. 3.3. En las figuras se pueden leer los nombres que fueron asignados a cada gesto para poder identificarlos.

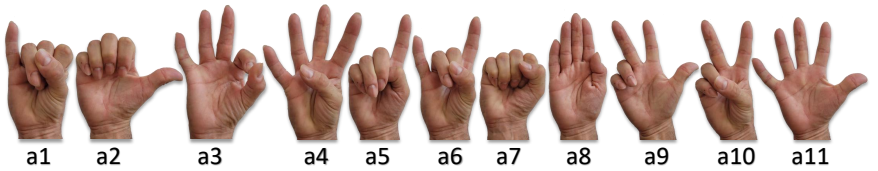


Figura 3.1: Gestos de [Memo et al., 2015]

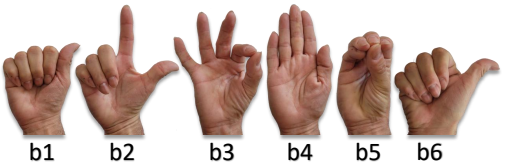


Figura 3.2: Gestos de [Bao et al., 2017]

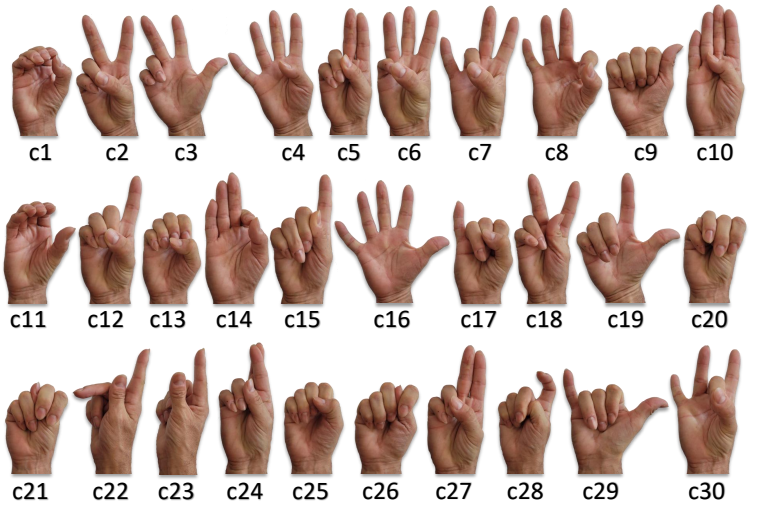


Figura 3.3: Gestos de [Thakur, 2019]

Ahora se busca caracterizar a una mano haciendo un gesto, es decir, determinar las cualidades y características de una mano para poder extraerlas como parámetros. En base a estas características, se pretende medir la cercanía entre los distintos gestos y seleccionar aquellos gestos que estén más alejados unos de otros. Los gestos en este

trabajo están determinados mediante el esqueleto de las manos, basado en 21 keypoints que están ubicados en las articulaciones; por este motivo, serán estos keypoints los que caracterizarán a una mano. En el esqueleto de la mano está implícita información de los ángulos definidos en cada articulación. El desafío está en establecer una manera de caracterizar a las manos, sabiendo que no necesariamente todos los dedos se flexionan en igual proporción o sabiendo que algunos de los dedos pueden estar flexionados y otros no, o algunos medianamente flexionados y otros totalmente extendidos. Hay diversas combinaciones y posibilidades en las posturas de los dedos de una mano que puede hacer difícil discernir entre los distintos gestos. Sabiendo esto, es posible medir el ángulo formado entre las rectas definidas por los keypoints de la mano en cada uno de sus dedos. Si, por ejemplo, consideramos el dedo mayor, se podrían medir los tres ángulos en las rectas formadas por los keypoints 0, 9, 10, 11 y 12, como se muestra en la Figura 3.4.

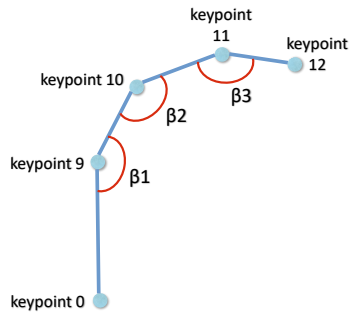


Figura 3.4: Ángulos del dedo mayor

Para realizar los cálculos de los ángulos entre estas rectas, se expresan todas las rectas en la forma paramétrica. Por ejemplo, en la Ec. 3.1 se muestra la expresión para la recta formada entre el keypoint 9 y el keypoint 10.

$$\begin{cases} x = kp9x + (kp10x - kp9x)\ \lambda \\ y = kp9y + (kp10y - kp9y)\ \lambda \\ z = kp9z + (kp10z - kp9z)\ \lambda \end{cases} \quad \text{con } \lambda \in \mathbb{R} \tag{3.1}$$

La expresión anterior es la forma paramétrica de la recta que pasa por los keypoints 9 y 10. Notar que si $\lambda = 0$ la expresión queda reducida al keypoint 9 y si $\lambda = 1$ se reduce al keypoint 10. Observando la Fig. 3.5, el cálculo de los ángulos se puede realizar como en la Ec. 3.2.

$$cos\ \alpha\ =\ \frac{\vec{r}\ .\ \vec{s}}{|\ \vec{r}\ |\ |\ \vec{s}\ |}\ =\ \frac{r_x\ s_x + r_y\ s_y + r_z\ s_z}{\sqrt{r_x^2 + r_y^2 + r_z^2}\ \sqrt{s_x^2 + s_y^2 + s_z^2}}$$

donde

$$\begin{aligned}\vec{r} &= (\ kp9x - kp0x,\ kp9y - kp0y,\ kp9z - kp0z\ ) \\ \vec{s} &= (\ kp10x - kp9x,\ kp10y - kp9y,\ kp10z - kp9z\ )\end{aligned} \tag{3.2}$$

El ángulo buscado es el suplementario de α, que es $\beta = (180° - \alpha)$

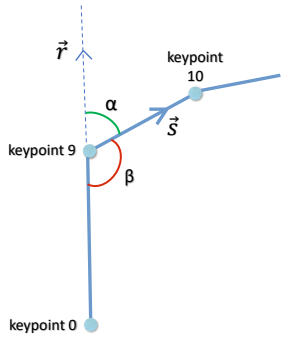


Figura 3.5: Cálculo de ángulos

Para realizar este análisis se capturaron 50 muestras por cada uno de los gestos de los 3 datasets seleccionados. El dataset [Memo et al., 2015] tiene 11 gestos, el dataset [Bao et al., 2017] tiene 6 gestos y el dataset [Thakur, 2019] tiene 30 gestos, haciendo un total de 47 gestos, por lo que se capturaron $47 \times 50 = 2\,350$ muestras. De cada una de estas muestras se extrajeron los keypoints y se calcularon los 15 ángulos por cada muestra, es decir, 15 ángulos por cada mano, 3 en cada dedo. Con el objetivo de simplificar esta caracterización, se pueden analizar los movimientos que se realizan con las manos y obtener la conjetura de que al flexionar un dedo, los 3 ángulos de las articulaciones son afectados, es decir, cuando se modifica un ángulo, se modifican todos, salvo que la persona tenga una habilidad especial para controlar el ángulo de una articulación de manera independiente sin afectar los otros 2 ángulos de ese dedo. Por este motivo, se considera aquí el sumatorio de los 3 ángulos de cada dedo para obtener un único valor por dedo que representará la flexión del mismo. Se realiza también una normalización de estos 5 valores que tiene cada mano, es decir, un valor de flexión en cada dedo. Cada uno de estos valores se deja acotado en el intervalo $[\,0,\,1\,]$. Por el hecho que la caracterización de una mano queda definida por 5 valores de flexión, resulta en un espacio multidimensional, de 5 dimensiones. Para que el método propuesto se pueda visualizar en gráficas, se lo explicará utilizando dos dimensiones, seleccionando el dedo índice y el mayor. En la Fig. 3.6 se muestra la posición en la que se ubican algunos de

los gestos en un gráfica en donde el eje x corresponde a la flexión del dedo índice, y el eje y la flexión del dedo mayor. Se puede observar que los gestos similares al puño, se encuentran cercanos al origen y cuando el dedo índice y mayor están extendidos, se ubican cercanos al $(x, y) = (1, 1)$. Notar que en la gráfica, cada montículo de distinto color corresponde a 50 muestras de un gesto particular. Como es de esperarse, se puede observar un montículo de muestras cercanas a $(x, y) = (0.5, 0.1)$ en donde el índice está con una flexión media, por ello los valores están cercanos al 0.5.

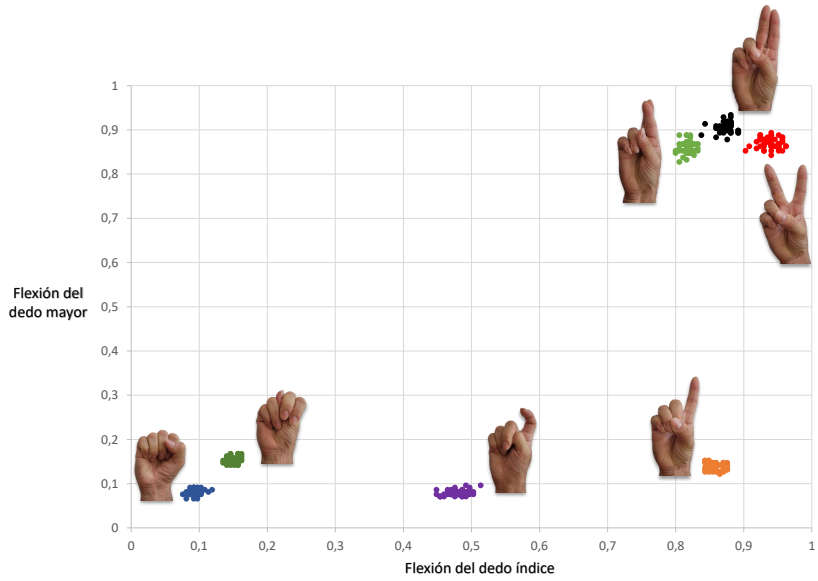


Figura 3.6: Flexión de los dedos índice y mayor

Ahora se realiza el agrupamiento de gestos que tengan similitud para poder seleccionar aquellos que sean discriminables entre ellos, y así favorecer al desempeño de las técnicas que se usen para la detección de gestos que se propone en esta tesis. El método de agrupamiento utilizado es k-means, el cual es un algoritmo no supervisado de agrupamiento (o clustering). El objetivo de este algoritmo es encontrar una cantidad k de grupos (o clusters) entre las muestras. k-means trabaja iterativamente para asignar cada muestra a uno de los k clusters. El agrupamiento se realiza minimizando la suma de distancias entre cada muestra y el centroide de su cluster. El algoritmo consta de tres pasos: 1) se inicializa eligiendo un k y estableciendo k centroides de manera aleatoria en el espacio de las muestras; 2) cada muestra es asignada a su centroide más cercano; 3) se actualiza la posición del centroide de cada cluster tomando como nuevo centroide la posición del promedio de las muestras pertenecientes a dicho cluster. k-means tiene la desventaja que es necesario saber de antemano cuántos clusters hay en las muestras para que el resultado sea aceptable. Sin embargo, en muchos casos no se conoce esta cantidad, y averiguarla, justamente puede ser la razón por la que desea realizar el clustering. El

conocimiento de las características de los datos puede ayudar a determinar el número de clusters. A pesar de ello, suponer el conocimiento de cuántas clases hay, va en contra de lo que es el aprendizaje no supervisado. Por ello, se necesita un método que indique el número de clusters. Un método disponible para esto, es Silhouette [Rousseeuw, 1987] o análisis de silueta, que estudia la distancia de separación entre los clusters resultantes. Silhouette utiliza un gráfico que visualiza una medida de la proximidad de cada muestra de un cluster a las muestras de los clusters vecinos y, por tanto, proporciona una forma de evaluar visualmente parámetros como el número de clusters. Esta medida tiene un rango de $[-1, 1]$, que son los coeficientes de silueta. Cuando estos coeficientes son cercanos a 1 indican que la muestra está alejada de los clusters vecinos. Un valor de 0 indica que la muestra está en la frontera de decisión entre dos clusters vecinos, y los valores negativos indican que esas muestras podrían haber sido asignadas al cluster equivocado. Existe suficiente literatura sobre k-means y Silhouette [de Amorim and Hennig, 2015], por lo que no se ampliará en esta tesis, sino que sólo se utilizarán para obtener los resultados y tomar las decisiones. El análisis de silueta para el ejemplo de los gestos, considerando únicamente el dedo índice y el mayor entrega como resultado una cantidad de 4 clusters, ubicados en el centroide de esas 4 agrupaciones, como se visualiza en la Fig. 3.7. Para este caso, una selección de 4 gestos sería la más adecuada, y serían: un gesto similar al puño; otro gesto con el dedo índice a flexión media y el dedo mayor contra la palma; otro con el índice totalmente extendido y el mayor contra la palma; y un cuarto gesto con el índice y mayor extendidos.

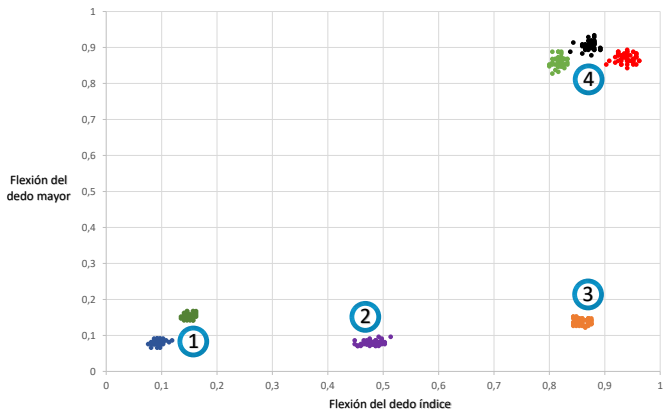


Figura 3.7: Clusters considerando dedo índice y mayor

Si se consideran las flexiones de los 5 dedos, no se podrá analizar mediante una gráfica en un sistema cartesiano, pero igualmente se podrá calcular la cantidad de clusters mediante el análisis de silueta. Tener presente que el análisis de silueta se prueba con todas las cantidades posibles de clusters y entrega como resultado un valor en el intervalo $[-1, 1]$ para cada cantidad de clusters, en donde un valor cercano a 1 indica que esa cantidad de clusters es una buena opción para llevar adelante la agrupación. Los resultados que entrega el análisis de silueta ordenados desde los valores más cercanos

a 1 es el 35 y el segundo mejor resultado es el 16. Teniendo estas dos cantidades de clusters en mente (35 clusters y 16 clusters), se puede realizar otro análisis utilizando la distancia euclidiana entre cada uno de los gestos. Como los gestos están caracterizados por un valor de flexión en cada dedo, esto hace que los datos tengan dimensión 5, y se puedan calcular estas distancias entre los 47 gestos disponibles. En la Fig. 3.8 se muestra un mapa de distancias en donde se visualizan las distancias entre un gesto y otro en base al color de cada celda, si el color es cercano al blanco, corresponde a una distancia muy pequeña, es decir, gestos similares, y si el color es un verde oscuro, la distancia es grande. Lo favorable para que un gesto sea fácilmente discriminable de otro y se pueda minimizar la confusión entre gestos, sería que los gestos estén alejados en distancia. En este mapa de distancias existe mucha información y es difícil tomar decisiones sin analizar una por una las distancias entre gestos. Se toma como ejemplo al gesto a11, que es la mano abierta, y se grafican los gestos con los cuales mayor distancia tiene, es decir, las que corresponden a las celdas más oscuras. Se puede observar que los gestos que mayor distancia tienen de una mano abierta son aquellos que forman un puño o similar. De esta manera se van analizando cada una de las distancias existentes entre los gestos y, junto con la información que entregó el análisis de silueta, seleccionar aquellos gestos que mejor se puedan discriminar.

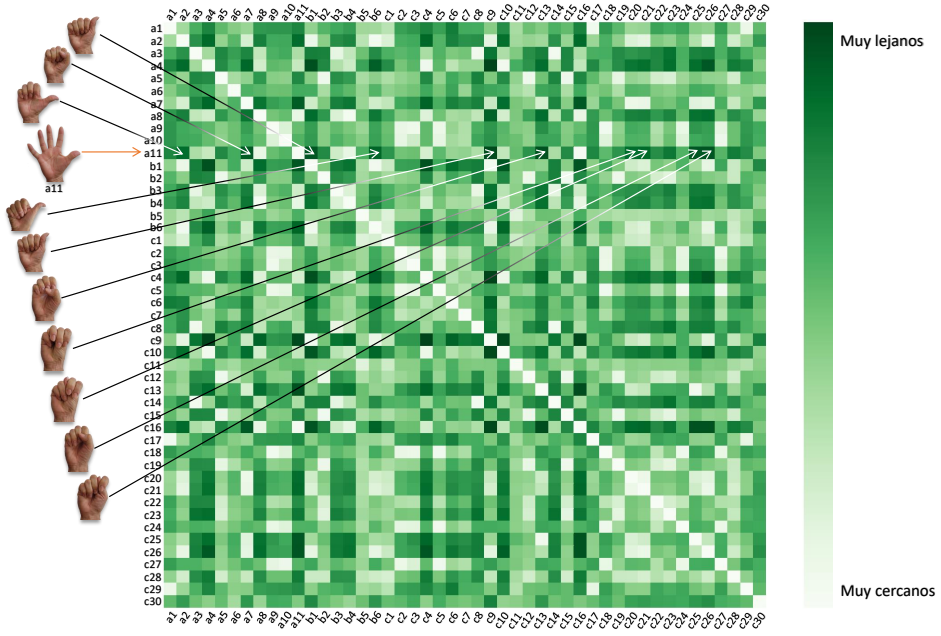


Figura 3.8: Mapa de distancias entre gestos

Recordando los dos mejores valores del análisis de silueta, la cantidad de 35 clusters corresponde a que existen varios gestos que son muy similares y están tan cercanos que

los agrupa en el mismo cluster, como los 12 gestos que se visualizan en la Fig. 3.9. Por este motivo, de un total de 47 gestos analizados, menos una cantidad de 12 gestos muy similares, se llega a 35 clusters. Estas 35 agrupaciones corresponden a 35 gestos que podrían ser correctamente discriminados entre ellos.

Figura 3.9: Pares de gestos muy similares

No obstante, para este trabajo de tesis se desea reducir la cantidad de gestos, por ello analizamos la agrupación en 16 clusters entregada por el análisis de silueta. Cada uno de estos 16 clusters tiene un centroide que agrupa en 16 clusters a los gestos con mayor similitud y que están a una distancia de los otros clusters que permitirían discriminarlos correctamente. En la Fig. 3.10 se muestran las agrupaciones que define k-means. En cada una de estas agrupaciones, cada gesto está clasificado de la misma manera, para los ojos de k-means es el mismo gesto. Sabiendo esto, y para que algún gesto sea el representativo de los demás gestos agrupados, en la gráfica se resalta en un tamaño mayor el gesto representativo de los demás en cada cluster.

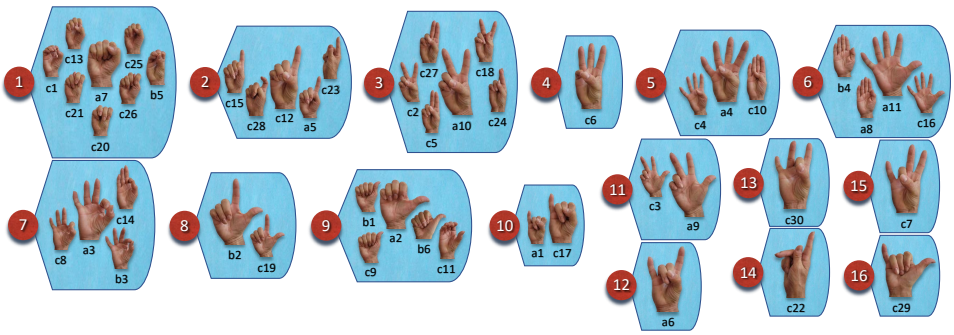


Figura 3.10: Gestos agrupados en 16 clusters

Con este análisis, es posible afirmar que la implementación de un algoritmo basado en los keypoints de las manos será efectiva para discriminar entre 16 gestos seleccionados. Entre los gestos a destacar se encuentran: el pulgar hacia arriba, ampliamente reconocido como señal de aprobación o acuerdo y tan popular que se ha convertido en un emoji habitual en redes sociales y valoraciones de servicios al cliente; el pulgar hacia abajo, que denota desaprobación; la V de Victoria o el signo de la Paz; el gesto OK, realizado al curvar el dedo índice hasta tocar el pulgar y extendiendo los otros dedos; y el gesto de la Corona, formado por el dedo índice y el meñique extendidos hacia arriba, mientras los dos dedos centrales se curvan hacia la palma, presionados por el pulgar, un gesto popular en la cultura del rock pesado y entre sus fanáticos, aunque también puede ser interpretado como un indicativo de infidelidad en la pareja. Cabe mencionar que no se

tomarán en cuenta gestos dinámicos, en los cuales la mano o alguno de sus dedos se muevan de una manera particular.

Respecto a la manera en que los investigadores realizan la selección de gestos para utilizar en sus proyectos, encontramos que en [Khan and Borji, 2018] utilizan el dataset EgoHands [Bambach et al., 2015], el cual tiene 16 posturas de manos, y los autores realizan una selección de 8 posturas según lo que ellos consideran más frecuentes. También se puede mencionar que la empresa tecnológica Snap Inc., con sus productos Snapchat (aplicación de mensajería con soporte multimedia de imagen, vídeo y filtros con realidad aumentada) y Spectacles (lentes de sol con cámara RGB integrada), deja disponible herramientas de desarrollo que permiten detectar 6 gestos que consideran más comunes, los cuales se visualizan en la Fig. 3.11.

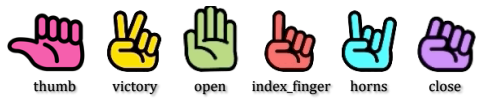


Figura 3.11: Gestos usados por Snap Inc.
Fuente: https://docs.snap.com/lens-studio/home

En el proceso de selección y análisis de gestos para esta tesis doctoral, se identificaron 16 gestos que presentaban características distintivas y reconocibles, con la confianza de que serían discriminables entre sí. Esta amplia selección inicial tenía como objetivo abarcar un espectro variado de gestos, de modo que se pudiera evaluar su potencial y aplicabilidad en el contexto del trabajo presente. Sin embargo, a lo largo del proceso de análisis, se hizo evidente que no todos los gestos inicialmente seleccionados serían idóneos para los objetivos específicos de esta investigación. Algunos de estos gestos, agrupados en los clusters 10, 13, 14 y 15, resultaron ser poco frecuentes en la comunicación no verbal cotidiana, lo que podría dificultar su realización correcta. Por lo tanto, se tomó la decisión de descartar estos gestos para simplificar el conjunto y focalizar la atención en aquellos gestos que serían más intuitivos y accesibles para los usuarios. Además, se identificaron gestos que, aunque eran claros y fácilmente realizables, podían generar ambigüedades con otros gestos del conjunto seleccionado. Este fue el caso del gesto del cluster 11, que se descartó para evitar confusiones con el gesto utilizado para indicar el número tres. Asimismo, se eliminó el gesto de la Corona (cluster 12) debido a las ambigüedades en su uso en la comunicación no verbal, y el gesto del cluster 16, comúnmente asociado con la realización de una llamada telefónica.

Al final, la decisión de reducir el conjunto de gestos de 16 a 9 se basó en la necesidad de garantizar la claridad, la facilidad de uso y la discriminabilidad entre los gestos, para asegurar el éxito de la aplicación práctica de estos en el marco de esta tesis doctoral. Los gestos finales seleccionados, correspondientes a los clusters del 1 al 9, fueron aquellos que cumplieron con todos estos criterios, proporcionando un conjunto robusto y eficaz para los propósitos de esta investigación.

Con el objetivo de identificar cada uno de estos gestos con un número o letra, se utilizan los signos extraídos de las siguientes lenguas de señas (visualizados en la Fig. 3.12):

- Desde International Sign Language: *1, 4, 5*
- Desde American Sign Language: *9, V, W*
- Desde French Sign Language: *A, L, S*

Figura 3.12: Nombres de los gestos

3.3. Creación del dataset

Para esta tesis fue creado un dataset propio a través de la grabación de videos de 10 voluntarios utilizando principalmente las cámaras RGB que vienen integradas en las computadoras portátiles y en algunos casos con las cámaras RGB de teléfonos móviles. Se les solicitó a cada voluntario que grabe un único video realizando los 9 gestos que se les indicó a través de un dibujo y un video ejemplo. Las grabaciones realizadas por los 10 voluntarios tuvieron una duración promedio aproximada de 3 minutos. Todos los movimientos de las manos para realizar los gestos fueron realizados con estilo libre, sólo guiados por el video ejemplo que se les envió como referencia. En la mayoría de los casos, la tasa de fotogramas de los videos fueron de 30 fps, por lo tanto, con 3 minutos de grabación por voluntario, y descartando los momentos del video en que alguno de los gestos no está presente, se dispone de gran cantidad de imágenes para la extracción de información útil para crear el dataset. Por cada voluntario se tiene un aproximado de 2 minutos y medio de video útil a 30 fps, lo que hace un total de 150 segundos $\times$ 30 $\frac{\text{fotogramas}}{\text{segundo}}$ = 4 500 fotogramas. Con estos números se puede decir que se tienen 500 imágenes por cada uno de los 9 gestos. Posteriormente, se realizó una edición de los videos de todos los voluntarios, seleccionando los momentos de cada gesto y uniendo en un video por separado cada gesto, obteniendo así 9 videos. Cada uno de estos videos contiene un gesto particular realizado por los 10 voluntarios y cada video tiene un aproximado de 4 500 fotogramas útiles. Finalmente, luego de la selección y edición de todos estos videos, no se obtuvo el total esperado de $4\,500 * 9 = 40\,500$ imágenes, porque algunos tuvieron que ser descartados, sino que se obtuvieron 39 150 imágenes de forma equilibrada entre gestos y voluntarios.

Para extraer la información de estas 39 150 imágenes, se desarrolló una herramienta en Python que implementa el modelo de *Mediapipe Hands* con la que se puede realizar la extracción de los 21 keypoints de las 39 150 imágenes. Es importante aclarar que estas imágenes se encuentran organizadas y etiquetadas por gesto, haciendo un total de 4 350 imágenes por gesto. En la Fig. 3.13 se muestran un par de estas imágenes siendo procesadas.

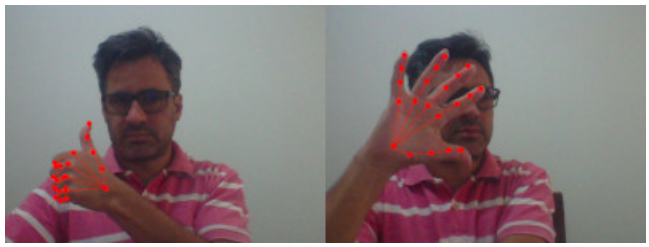


Figura 3.13: Ejemplo de las imágenes para extracción de keypoints

El dataset generado consiste de un archivo CSV (comma-separated values) conteniendo 39 150 muestras (4 300 por cada gesto) con la información que se muestra en la Fig. 3.14. Cada muestra tiene 64 columnas de información: el nombre del gesto más los 21 keypoints (x, y, z). Este dataset fue dividido en una proporción de 80 % (31 320 muestras) para entrenamiento y validación (train/val) y 20 % (7 830 muestras) para pruebas (test).

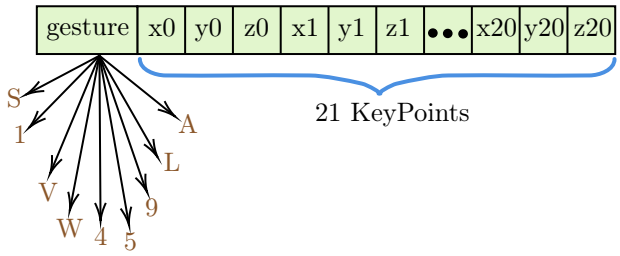


Figura 3.14: Información almacenada

Adicionalmente se han descargado 3 datasets externos [Thakur, 2019], [Memo et al., 2015] y [Bao et al., 2017] para disponer de muestras que permitan comprobar el desempeño del modelo que se propone en esta tesis con imágenes totalmente ajenas. Vale aclarar que estos datasets externos no disponen cada uno de los 9 gestos que son de interés aquí. Por ello se realizó una combinación de gestos de estos datasets y se realizó el mismo trabajo de selección y edición para finalmente obtener un total de 4 500 muestras para test (500 por cada gesto).

3.4. Normalización de datos

La normalización se aplica a menudo como parte de la preparación de un dataset para que los algoritmos de machine learning brinden mejores resultados. El objetivo de la normalización es modificar los valores de los datos para usar una escala común, sin distorsionar los intervalos de valores ni perder información. Si se supone, por ejemplo, que existe un conjunto de datos que tiene valores entre 10 000 y 90 000, y otro conjunto de datos con valores entre 0 y 1. Esta gran diferencia de escala puede ocasionar problemas cuando se extraigan características y patrones durante el entrenamiento del modelo. En este ejemplo se puede aplicar un cambio de escala y llevar todos los valores al rango entre 0 y 1, o trabajar con porcentajes en lugar de valores absolutos. En el caso de los valores de los keypoints almacenados en el dataset propio, la normalización también puede ser del tipo escalado, ya que, algunas manos más pequeñas o más alejadas a la cámara, resultan en magnitudes menores.

La normalización elegida para los keypoints almacenados en el archivo CSV consiste en varias transformaciones del tipo traslación, rotación y escalado, que permitan ubicar los keypoints de cada mano en el origen del espacio tridimensional y que el dedo mayor quede alineado con el eje Y.

Las matrices utilizadas para lograr esta normalización, son las siguientes:

- Matriz (3.3) de traslación al origen.

$$T = \begin{bmatrix} 1 & 0 & 0 & -kp0x \\ 0 & 1 & 0 & -kp0y \\ 0 & 0 & 1 & -kp0z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

  donde el keypoint 0 es $(x, y, z) = (kp0x, kp0y, kp0z)$

- Matriz de rotación (3.4) sobre un eje arbitrario: Para alinear el dedo mayor con el eje Y.

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

  donde:

$$r_{11} = cos\ \theta + u_x^2(1 - cos\ \theta)$$
$$r_{12} = u_x u_y(1 - cos\ \theta) - u_z sin\ \theta$$

$$
\begin{aligned}
r_{13} &= u_x u_z(1 - cos\ \theta) + u_y sin\ \theta \\
r_{21} &= u_y u_z(1 - cos\ \theta) + u_z sin\ \theta \\
r_{22} &= cos\ \theta + u_y^2(1 - cos\ \theta) \\
r_{23} &= u_y u_z(1 - cos\ \theta) - u_x sin\ \theta \\
r_{31} &= u_z u_x(1 - cos\ \theta) - u_y sin\ \theta \\
r_{32} &= u_z u_y(1 - cos\ \theta) + u_x sin\ \theta \\
r_{33} &= cos\ \theta + u_z^2(1 - cos\ \theta) \\
r_{33} &= cos\ \theta + u_z^2(1 - cos\ \theta)
\end{aligned}
$$

Aquí, u es el vector unitario, el cual es perpendicular al plano formado por el vector keypoint 9 y el eje Y. Sabiendo que estos 2 vectores son perpendiculares (u ortogonales) cuando su producto punto (o producto escalar) es igual a cero, entonces es posible calcular el vector u. O también es posible utilizar el producto vectorial (o producto cruz) entre el keypoint 9 y el eje Y. Para esto, se puede utilizar la Regla de Sarrus para calcular el determinante 3×3 y así obtener un vector perpendicular al plano entre el keypoint 9 y el eje Y. Finalmente se lo divide por la norma para obtener el vector unitario u de la expresión previa.

Por Regla de Sarrus se obtiene un vector (Ec. 3.5) que, en general, no es unitario. Considerar que el vector en el eje Y es unitario, es decir, es el vector $(0, 1, 0)$:

$$
\begin{aligned}
u'_x &= +(kp9y \times 0 - 1 \times kp9z) = -kp9z \\
u'_y &= -(kp9x \times 0 - 0 \times kp9z) = 0 \\
u'_z &= +(kp9x \times 1 - 0 \times kp9y) = +kp9x
\end{aligned}
\tag{3.5}
$$

Cuando se divide por su norma, obtenemos el vector unitario u, como se muestra en la Ec. 3.6.

$$
\begin{aligned}
\mid u' \mid &= \sqrt{(-kp9z)^2 + 0 + (kp9x)^2} \\
u &= \left(\frac{-kp9z}{\mid u' \mid}, 0, \frac{kp9x}{\mid u' \mid} \right)
\end{aligned}
\tag{3.6}
$$

θ es el ángulo entre el vector formado desde el origen hasta el inicio del dedo mayor (es decir, el keypoint 9) y el vector unitario sobre el eje Y. Esto puede ser calculado con la Ec. 3.7.

$$
\theta = cos^{-1} \left(\frac{kp9y}{\sqrt{(kp9x)^2 + (kp9y)^2 + (kp9z)^2}} \right) \tag{3.7}
$$

- Matriz (3.8) para rotar la palma en el eje Y para alinearlo al plano $z = 0$.

$$R_y = \begin{bmatrix} cos\ \beta & 0 & sin\ \beta & 0 \\ 0 & 1 & 0 & 0 \\ -sin\ \beta & 0 & cos\ \beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

  β es el ángulo sobre el plano $y = 0$ del ángulo formado entre el keypoint 17 y el eje X. De esta manera se alinea la palma con el plano $z = 0$ como muestra la Ec. 3.9.

$$\beta = tan^{-1}\ \left(\frac{kp17z}{kp17x}\right) \tag{3.9}$$

- Matriz de rotación sobre el eje Y para ubicar la palma de manera frontal: se usa la matriz R_y para rotar 180° sobre el eje Y siempre y cuando la palma se encuentre en dirección de los valores negativos de z. Para saber si la palma se encuentra de frente o no, un cálculo simple se puede realizar para detectar hacia dónde se orientan la punta de los dedos.

- Reflexión respecto al plano $x = 0$: Independientemente de si se trata de la mano derecha o la izquierda, queremos reflejar la mano de tal manera que el pulgar siempre permanezca hacia los valores positivos de x. Es simple detectar si el pulgar está a la derecha o a la izquierda averiguando los valores x de los keypoints pertenecientes al pulgar.

- Escalado: La mano es escalada de forma tal que la magnitud de $\mid kp0y - kp9y \mid$ sea igual a 100. La matriz (3.10) es usada para realizar dicho escalado.

$$E = \begin{bmatrix} \frac{100}{kp9y} & 0 & 0 & 0 \\ 0 & \frac{100}{kp9y} & 0 & 0 \\ 0 & 0 & \frac{100}{kp9y} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

Para obtener los valores normalizados, las operaciones con las matrices de la Ec. 3.11 deben ser realizadas con cada uno de los 21 keypoints de cada mano.

$$\begin{bmatrix} x_{\mathrm{n}} \\ y_{\mathrm{n}} \\ z_{\mathrm{n}} \\ 1 \end{bmatrix} = E\ R_y\ R\ T \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.11}$$

donde x_n, y_n y z_n son las coordenadas de los keypoints normalizados.

Hay que tener en cuenta que, según el caso, también se realiza el giro de 180° y/o el espejado con respecto al plano $x = 0$.

Para llevar a cabo la normalización de los keypoints, se ha desarrollado una herramienta que permite visualizar la correcta normalización de los keypoints que componen el dataset. En la Fig. 3.15, se muestra una mano en su posición original y en la Fig. 3.16 se visualiza después de la normalización. Algunos keypoints fueron unidos con líneas para una clara visualización del esqueleto de la mano.

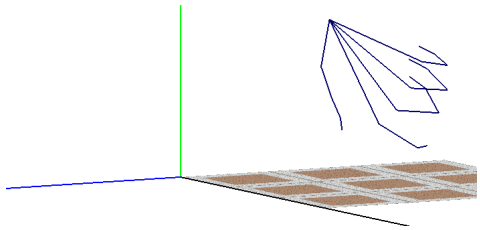


Figura 3.15: Esqueleto de una mano en su posición original

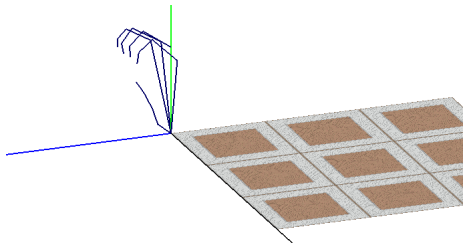


Figura 3.16: Esqueleto de una mano después de normalizar

3.5. Arquitectura de la red PointNet

La red neuronal PointNet [Qi et al., 2017] recibe en su capa de entrada una nube de puntos para la clasificación de objetos. A continuación se exponen tres características que son afirmativas cuando se clasifica una nube de puntos:

- Invarianza a la permutación: una nube de puntos es un conjunto de datos en bruto, sin información adicional. Es una colección de coordenadas (x, y, z) sin estructura. Esto hace que los datos sean invariables a las permutaciones.
- Invarianza a ciertas transformaciones: la clasificación de los objetos no debe cambiar si la nube de puntos sufre transformaciones de traslación y/o rotación, no así con el escalado.
- Importancia entre puntos vecinos: cada punto no se trata de forma independiente, ya que la interacción entre puntos vecinos contiene información útil.

Es importante señalar que es habitual considerar que una nube de puntos tiene un gran número de puntos. Los autores de PointNet utilizaron en su trabajo una nube de 2 048 puntos para cada objeto, utilizando el conjunto de datos ModelNet10 [Zhirong Wu et al., 2015], que contiene objetos pertenecientes a 10 clases. Esta red toma n puntos de entrada, cada uno con dimensión 3 perteneciente a las coordenadas (x, y, z), por lo que tiene una entrada con dimensión $[2\,048, 3]$. PointNet tiene distintas capas que realizan transformaciones sobre los datos sin modificar su dimensión. Estas capas están compuestas por convoluciones unidimensionales o temporales (Conv1D) con activación ReLU, Max Pooling y capas densamente conectadas (Fully Connected).

Aquí se hace una breve explicación de las diferencias entre las operaciones convolucionales unidimensionales o temporales (Conv1D) y las convoluciones bidimensionales o espaciales (Conv2D) explicadas en la sección 2.2.5. Cuando se habla de redes neuronales convolucionales, generalmente se hace referencia a las convoluciones bidimensionales o espaciales que se utilizan para la clasificación de imágenes. Pero también son utilizadas las redes convolucionales unidimensionales e incluso las tridimensionales. Ejemplos en los que se utilizan estos tres tipos de redes convolucionales pueden ser: Conv1D para datos de series de tiempo, para señales, para datos recolectados por sensores; Conv2D para imágenes; Conv3D para imágenes 3D como la resonancia magnética que se utiliza para examinar el cerebro y los órganos internos del cuerpo humano y la tomografía computarizada que generan datos tridimensionales a partir de la toma de imágenes en diferentes ángulos del cuerpo con rayos X.

En una Conv1D, el kernel se mueve en una única dirección (Fig. 3.17) . En Conv2D, el kernel se mueve en dos direcciones (ver Fig. 3.18 que muestra la convolución de un kernel sobre una imagen de 3 canales). El Conv3D se mueve en tres direcciones (Fig. 3.19).

Después de distintas capas combinadas con otras capas de convolución, se realiza un Max Pooling tomando el valor máximo global de los datos, disminuyendo la dimensionalidad. Le siguen capas Fully Connected y una última capa con función de activación softmax para obtener las puntuaciones de k clases de salida.

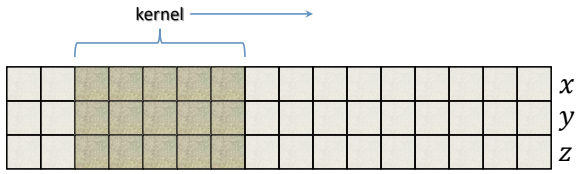


Figura 3.17: Ejemplo de convolución unidimensional o temporal

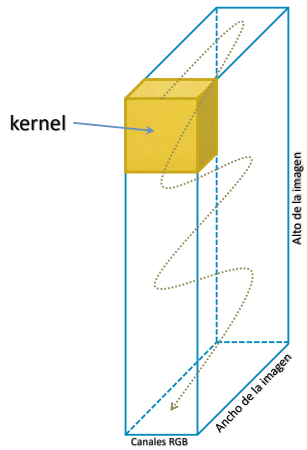


Figura 3.18: Ejemplo de convolución bidimensional o espacial sobre imágenes

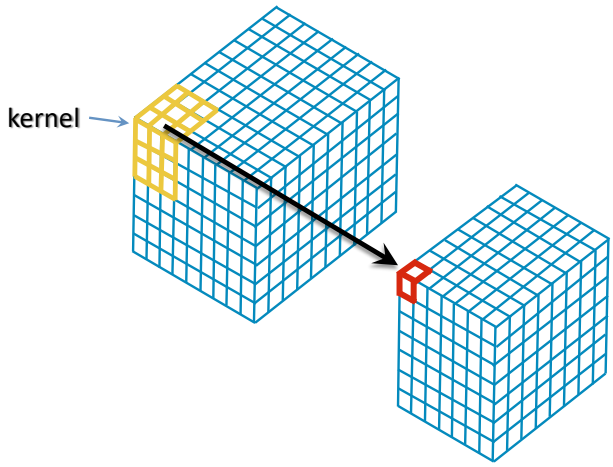


Figura 3.19: Ejemplo de convolución tridimensional o espacial sobre volúmenes

3.6. Arquitectura de red propuesta

Para definir la arquitectura de red se toma como base a la red PointNet, la cual tiene 19 capas ocultas más la capa de salida con función softmax. PointNet realiza transformaciones afines en los datos y se propone eliminarlas, ya que las muestras del dataset propio se someten a las operaciones de transformación y normalización que fueron definidas en la sección 3.4. Se realizaron diversas pruebas y se fue evaluando el desempeño de las arquitecturas elaboradas. Las pruebas consistieron en reducir la cantidad de capas y emplear diferentes cantidades de neuronas, evaluando las métricas durante el entrenamiento y la tasa de aciertos con el modelo generado. Se fue reduciendo considerablemente la cantidad de capas ocultas, esperando una disminución en la tasa de aciertos, aunque no sucedió hasta llegar a dos capas ocultas. Al reducir la cantidad de capas se produce una disminución en el tiempo de entrenamiento, el tamaño del modelo ocupa menos espacio en memoria y también disminuye el tiempo de procesamiento con el modelo generado. La arquitectura quedó definida con 3 capas ocultas más la capa de salida con softmax y se muestra en la Fig. 3.20. Hay que tener en cuenta que la normalización de los datos explicada anteriormente se aplica a las muestras antes de ingresar a la red, por lo que no se visualiza en esta figura.

	Activation Shape	# parameters
Input	(21, 3)	0
Conv1D	(21, 32)	128
BatchNormalization	(21, 32)	128
Activation	(21, 32)	0
Conv1D	(21, 64)	2112
BatchNormalization	(21, 64)	256
Activation	(21, 64)	0
GlobalMaxPooling1D	(64, 1)	0
Dense	(128, 1)	8320
BatchNormalization	(128, 1)	512
Activation	(128, 1)	0
Dropout	(128, 1)	0
Softmax	(9, 1)	1161

Cantidad de parámetros = 12 617 → Entrenables = 12 169
No entrenables = 448

Figura 3.20: Arquitectura de red propuesta

A continuación se realiza un análisis de las distintas capas de la red propuesta:

- InputLayer: Es la capa de entrada con los 21 keypoints de la mano y cada keypoint tiene 3 dimensiones correspondiente a las coordenadas (x, y, z).

- Capa oculta 1: Esta primera capa está compuesta por una convolución temporal con 32 kernels unidimensionales de tamaño 1 con normalización por lotes y activación ReLU. Esta capa tiene en total 256 parámetros, de los cuales 192 son entrenables.

  - Conv1: En la Fig. 3.21 se muestra la aplicación del kernel en una convolución unidimensional, es decir, que el kernel se desplaza en una única dirección. El kernel tiene tamaño 1 y debido a que los datos de entrada poseen 3 dimensiones, entonces el kernel debe tener 3 valores más el bias. Por lo tanto, cada kernel tiene 4 parámetros entrenables y como se aplican 32 kernels en esta primera capa oculta, se tiene un total de 128 parámetros entrenables. Luego de las operaciones con los 32 kernels, la salida tendrá dimensión 21×32 como muestra la Fig. 3.22.

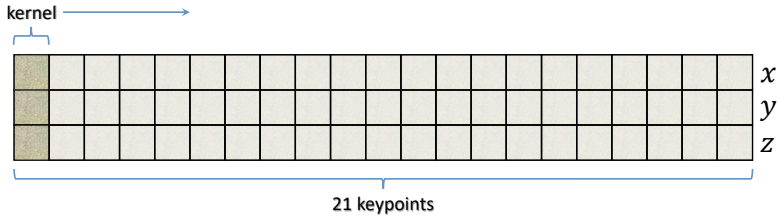


Figura 3.21: Kernel Conv1D aplicado a los 21 keypoints

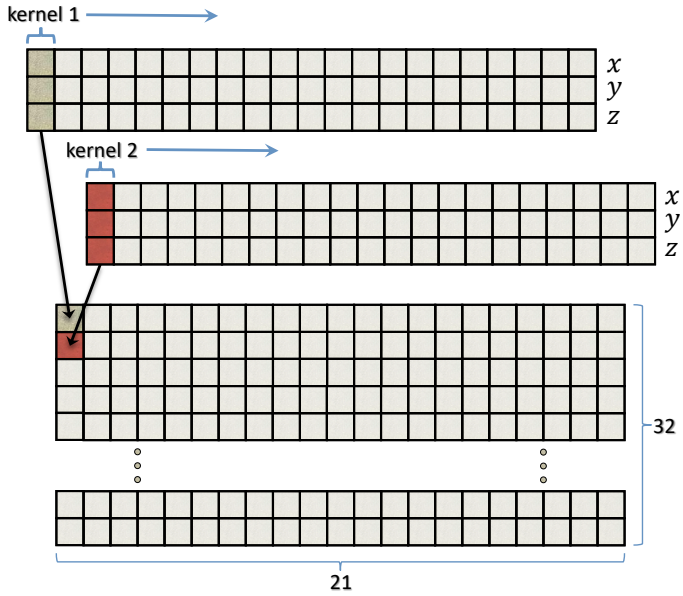


Figura 3.22: Salida de la primera convolución Conv1D

• BatchNormalization: A la hora de crear un dataset se tiene la posibilidad de realizar la normalización de los datos, pero una vez que los datos son ingresados a la red neuronal puede ser necesario volver a normalizar los datos, ya que entre capa y capa puede haber un desajuste de escala. Para poder insertar una operación de normalización entre cada capa se tiene la normalización de lote[1] (o batch normalization) que es un método que normaliza la salida de una capa restando la media de los valores de salida y dividiendo por la desviación estándar del mismo lote. Al realizar esta normalización los pesos de la capa siguiente no son óptimos para estos valores de entrada, para ello BatchNormalization entrena dos parámetros, que son la desviación estándar γ que multiplica a la salida y la media β que se le suma, como muestran las Ec. 3.12. Además de estos dos parámetros entrenables (γ y β), el BatchNormalization posee otros dos parámetros para almacenar las actualizaciones de la media y la varianza cada vez que se invoca a este algoritmo, pero son parámetros no entrenables. Entonces, esta primera BatchNormalization posee un total de 128 parámetros (64 entrenables y 64 no entrenables).

Entrada: (3.12)

Datos del mini lote: $\mathbb{B} = \{\ x_1,\ x_2,\ \ldots\ ,\ x_m\ \}$

Parámetros para aprender: $\gamma\ ,\ \beta$

Salida:

$y_i = \mathrm{BN}_{\gamma,\beta}\ (x_i)$ $\qquad$ BN = Batch Normalization

donde:

Media del mini-batch: $\mu_{\mathbb{B}} = \dfrac{1}{m}\ \displaystyle\sum_{i=1}^{m} x_i$

Varianza del mini-batch: $\sigma^2_{\mathbb{B}} = \dfrac{1}{m}\ \displaystyle\sum_{i=1}^{m} (x_i - \mu_{\mathbb{B}})^2$

Normalización: $\hat{x_i} = \dfrac{x_i - \mu_{\mathbb{B}}}{\sqrt{\sigma^2_{\mathbb{B}} + \epsilon}}$ $\quad \epsilon \approx 0$ para evitar división por cero

Salida: $y_i = \gamma\ \hat{x_i} + \beta = \ \mathrm{BN}_{\gamma,\beta}\ (x_i)$

• Activation: Aplica la función de activación ReLU definida por la expresión $f(x_i) = \max(\ 0,\ x_i\ )$. Aquí no hay parámetros para el aprendizaje.

[1]Cuando se habla de lote, se hace referencia al entrenamiento por lotes (o batch) o también entrenamiento por mini lotes (o mini batch). En el entrenamiento por lotes, se actualizan los pesos luego de pasar por todas las muestras de entrenamiento del dataset. Y en el entrenamiento por mini lotes, se dividen las muestras de entrenamiento en grupos de menor tamaño y se actualizan los pesos tras pasar por cada grupo.

- Capa oculta 2: Esta segunda capa utiliza Conv1D con 64 kernels, con normalización por lotes y activación ReLU. Los datos de salida de la capa oculta 1 tienen dimensión de 21×32 por lo que, aplicar un kernel de tamaño 1, entonces este kernel debe tener 32 parámetros más el bias ($33 \times 64 = 2\,112$ parámetros entrenables). La salida de Conv1D tiene dimensión 21×64, entonces el BatchNormalization tiene un total de 256 parámetros (128 entrenables y 128 no entrenables).

- GlobalMaxPooling1D: Realiza un max pooling tomando el máximo valor de los 21 valores en cada fila, es decir, la entrada a este max pooling es 21×64 y la salida es sólo de 64.

- Capa oculta 3: Esta es la tercer capa oculta de la red y es del tipo Fully Connected, con normalización por lotes y activación ReLU. La entrada a esta capa es de dimensión 64 más el bias y si se utiliza conexión densa, entonces se disponen de $65 \times 128 = 8\,320$ pesos entrenables. BatchNormalization con 256 parámetros entrenables más 256 no entrenables.

- Dropout: La operación de dropout desactiva las contribuciones de ciertas neuronas de manera aleatoria durante las iteraciones del entrenamiento. Por ejemplo, en la Fig. 3.23 se puede observar una pequeña red Fully Connected en la cual se desactivan ciertas neuronas durante alguna iteración de manera que no aporten al resultado final. En la próxima iteración se desactivarán otras neuronas distintas. De esta manera se generaliza para que las neuronas no dependan unas de otras. Es muy utilizado para evitar el overfitting (o sobreentrenamiento[2]) y se utiliza con un valor que indica el porcentaje de neuronas que se desactivarán en cada iteración. En esta red propuesta se desactiva un 30 % de las neuronas. En el dropout no hay parámetros entrenables.

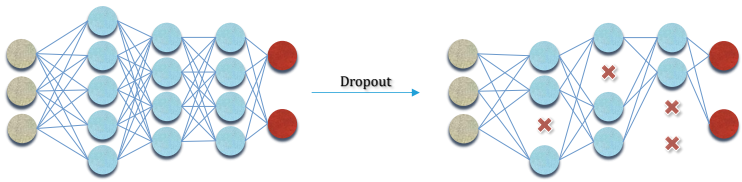


Figura 3.23: Dropout

- Capa de salida: Fully Connected con función softmax con 9 neuronas de salida para la clasificación de los 9 gestos de las manos. La entrada a esta útlima capa es de dimensión 128 más el bias y como se utiliza conexión densa, entonces se disponen de $129 \times 9 = 1\,161$ pesos entrenables. Si se tienen k clases diferentes para un problema de clasificación, se utilizan k neuronas de salida. Softmax permite interpretar los niveles de activación de las neuronas de salida como estimaciones de la probabilidad de que un ejemplo pertenezca a cada una de las clases.

[2]El sobreentrenamiento se ve reflejado cuando el modelo entrenado aprende tanto a distinguir los datos del dataset que es incapaz de reconocer nuevos datos de entrada.

Esta arquitectura de red posee un total de 12 169 parámetros entrenables que se ajustarán durante el entrenamiento utilizando el 80 % de las muestras totales del dataset creado (31 320 muestras). El resto (7 830 muestras) será utilizado para realizar las pruebas con el modelo entrenado y adquirir las métricas de desempeño.

3.7. Data augmentation

Data augmentation es la generación artificial de datos por medio de modificaciones en los datos originales. Permite agregar mayor cantidad de datos y diversidad al dataset para realizar el entrenamiento. Es una técnica muy utilizada para los datasets de imágenes a las que se les aplican rotaciones, se amplían regiones, se voltean, se les cambia el brillo, se recortan, se eliminan pequeñas regiones, entre otras. Teniendo en cuenta que un dataset de keypoints de las manos no cuenta con imágenes sino con coordenadas tridimensionales, las transformaciones que se pueden aplicar son distintas. Aquí se propone incrementar la cantidad de puntos para cada mano.

Al analizar una gráfica de los 21 keypoints de una mano puede ser difícil, para el ojo humano, identificar a qué gesto corresponden esos puntos. Se puede considerar que 21 keypoints son insuficientes para representar el esqueleto de una mano, por lo que se propone generar datos extras sabiendo que entre ciertos keypoints hay un hueso (en la palma los huesos metacarpianos, en el principio de los dedos las falanges proximales, seguidas de las falanges medias y en la punta de los dedos las falanges distales). A este respecto, se define un nuevo parámetro que permite incorporar una cierta cantidad de keypoints adicionales en los huesos de la mano. Esto se consigue calculando las líneas que unen los keypoints que corresponden a los extremos de los huesos mencionados anteriormente. En la Fig. 3.24 se visualizan los keypoints de una mano con la adición de 10 keypoints en cada hueso.

Figura 3.24: Mano con 10 keypoints extra en cada hueso

Para esta tesis se desarrolló una herramienta que permite agregar la cantidad de puntos que se deseen. La información de entrada de esta herramienta es el número de puntos extras por cada hueso.

3.8. Entrenamiento

La arquitectura de la red propuesta fue implementada con la biblioteca Keras y se dispone de 31 320 muestras para el entrenamiento/validación. Vale mencionar que durante el entrenamiento se realiza una validación del aprendizaje del modelo a medida que se va avanzando en las iteraciones del entrenamiento. Para ello se utiliza una porción de los datos destinada a la validación. En esta propuesta se elige un 80 % para el entrenamiento y un 20 % para la validación. Notar que esta separación de muestras es sobre las 31 320 muestras, totalmente separadas de las 7.830 muestras destinadas para la prueba del modelo luego de haberlo entrenado. Tener presente que también se dispone de un conjunto de muestras extraído de imágenes de terceros y que será utilizado también para las pruebas del modelo. Las proporciones quedan así:

- Total de muestras del dataset propio: 39 150
- Subconjunto para entrenamiento: 25 056 $=$ *80 % de* 31 320
- Subconjunto para validación: 6 264 $=$ *20 % de* 31 320
- Subconjunto para pruebas: 7 830
- Total de muestras de datasets de terceros: 4 500

Luego de tener disponible la arquitectura de red, es necesario configurar el proceso de aprendizaje con los siguientes tres parámetros principales:

- Función de pérdida (o función de coste): La función de coste pretende determinar el error entre el valor estimado y el valor real, con el objetivo de optimizar los parámetros de la red neuronal. En esta propuesta se utiliza la función de entropía cruzada categórica, que es frecuentemente usada en problemas de clasificación cuando se utiliza softmax en la capa de salida. Para desmembrar este concepto, primero se puede definir a la *entropía* como una medida de incertidumbre. Cuando todas las clases están igualmente distribuidas y no es posible decidirse por una, entonces la incertidumbre es total, por lo que la entropía es máxima. Al reducir la entropía se gana información, lo que permite distinguir una clase por sobre otras. La *entropía cruzada* compara la probabilidad de cada clase devuelta por softmax con respecto a la clase real, calculando la pérdida de manera que penaliza la probabilidad según la lejanía que tenga con el valor esperado. Esta pérdida es utilizada para ajustar los pesos de la red durante el entrenamiento. Mientras mejor sea un modelo, su entropía tiende a 0. La *entropía cruzada categórica* es una ampliación de la función de entropía cruzada, adaptada a problemas en los que se tienen que clasificar los datos en más de dos clases.

- Método de optimización: Las redes neuronales modelan el problema (por ejemplo, problema de clasificación) en base a unos parámetros, también conocidos como pesos, correspondientes a la representación matemática interna de la red.

Entre cada capa se tiene una matriz de pesos. El optimizador busca reducir el error cometido por la red utilizando un proceso conocido como backpropagation. Esta actualización tiene en cuenta la derivada de las matrices de pesos. El optimizador más simple actualiza los valores de los pesos, equitativamente, según la tasa de aprendizaje (learning rate). Dicho de otra manera, para agilizar la convergencia de la función de coste hacia su mínimo, se multiplica el vector de gradiente por el learning rate. Entre los métodos de optimización más utilizados se encuentran: Descenso de Gradiente Estocástico (SGD: Stochastic Gradient Descent), RMSprop (Root Mean Square Propagation) y Adam (Adaptive moment estimation). En el descenso de gradiente (no estocástico), todos los datos de entrenamiento se introducen de una vez y el gradiente se calcula usando todas las muestras. En el descenso de gradiente estocástico se introduce una o más muestras aleatorias en cada iteración para realizar los cálculos del gradiente. El learning rate en el descenso de gradiente estocástico es fijo. En el método RMSprop y Adam, se introduce una variación muy interesante que, en lugar de mantener el learning rate único y fijo para todos los pesos, lo adapta a cada peso a medida que avanza el entrenamiento. Todas estas mejoras buscan una convergencia más rápida de la función de coste. Adam es un algoritmo popular en el campo del aprendizaje profundo porque los resultados empíricos demuestran una rápida convergencia y se compara favorablemente respecto a otros métodos de optimización al entrenar una red del tipo perceptrón multicapa [Kingma and Ba, 2015]. Debido a ello, en esta propuesta de tesis se utiliza el optimizador Adam.

- Métricas: Las métricas permiten monitorear el entrenamiento del modelo. Las métricas se registran al final de cada época, tanto para el conjunto de datos de entrenamiento como para el conjunto de datos de validación. La métrica utilizada durante el entrenamiento es la exactitud (accuracy) categórica dispersa (Sparse Categorical Accuracy) que calcula la tasa con la que las predicciones coinciden con las etiquetas verdaderas.

Para realizar el entrenamiento quedan por seleccionar los hiperparámetros y generar el modelo que posteriormente servirá para realizar las predicciones dentro de desarrollos que requieran la detección de gestos de las manos. En esta propuesta, el modelo generado tiene un formato HDF5 (Hierarchical Data Format Release 5) que permite almacenar grandes conjuntos de datos numéricos[3].

[3]HDF5 - `https://www.hdfgroup.org/solutions/hdf5`

Capítulo 4

Entrenamiento y evaluación del modelo

Capítulo 4

Entrenamiento y evaluación del modelo

Contenidos

4.1. Entrenamiento

Para realizar el entrenamiento del modelo se utilizó la herramienta Google Colaboratory[1] que permite ejecutar scripts de Python a través de los servidores de Google. Se implementó la arquitectura de la red neuronal junto a un script que permite realizar cíclicamente distintos entrenamientos y validaciones del modelo generado con las variaciones de sus hiperparámetros, tal como la cantidad de épocas, la tasa de aprendizaje, el número de keypoints extras por cada hueso, entre otros. Los hiperparámetros que se utilizaron fueron los siguientes:

- Tasa de aprendizaje: *0.0001, 0.0005, 0.001, 0.005, 0.01*
- Épocas: *10, 20, 30, 50, 100*
- Keypoints extras en cada hueso: *0, 1, 2, 5, 10, 20, 50*

De esta manera se realizaron más de 100 procesos de entrenamiento junto a sus métricas de evaluación del modelo obtenido. En la Tabla 4.1 se muestran los mejores resultados ordenados según la tasa de aciertos en las predicciones realizadas con los datos de prueba. Se dispone de dos conjuntos de datos de prueba: por un lado, los keypoints pertenecientes a las 7 830 muestras extraídas del dataset propio y, además, 4 500 muestras de los datasets externos. Esta tabla de resultados muestra lo siguiente: cantidad de keypoints adicionales añadidos en cada hueso; la tasa de aprendizaje del optimizador Adam; la cantidad de épocas para el entrenamiento con un tamaño de lote de 32 con una división de 80 %/20 % para el entrenamiento/validación; el total de keypoints para cada mano; la pérdida en el conjunto de entrenamiento después de todas las épocas; la exactitud (accuracy) con el conjunto de entrenamiento; la pérdida en el conjunto de validación después de todas las épocas; la exactitud (accuracy) en el conjunto de validación; la tasa de éxito en las predicciones realizadas con el conjunto de prueba de 4 500 muestras externas; y la tasa de éxito en las predicciones realizadas con el conjunto de prueba de 7 830 muestras propias. El tiempo consumido para realizar cada predicción es, en promedio, de 26 milisegundos, en una computadora portátil Dell Inspiron 5502 con procesador Intel Core™ i7-1165G7 11^a Generación con 4 núcleos (sin GPU), que equivale a un promedio de 38 fps.

[1] Google Colaboratory (o Google Colab - `https://colab.research.google.com`) es muy útil para implementar algoritmos de machine learning utilizando recursos de hardware como GPU para potenciar el cómputo.

#	Extra KeyPoints on each bone	Learning rate	Epochs	Total KeyPoints	training loss	training acc	validation loss	validation acc	Correct predictions (ext dataset)	Correct predictions (%) (ext dataset)	Correct predictions (own dataset)	Correct predictions (%) (own dataset)
1	0	0.0005	10	21	0.0068	99.84 %	0.0048	99.95 %	4336 of 4500	96.36 %	7820 of 7830	99.87 %
2	2	0.0005	30	61	0.0085	99.80 %	0.0056	99.89 %	4305 of 4500	95.67 %	7820 of 7830	99.87 %
3	10	0.001	50	221	0.0062	99.85 %	0.0094	99.89 %	4297 of 4500	95.49 %	7820 of 7830	99.87 %
4	0	0.0005	30	21	0.0055	99.88 %	0.0072	99.86 %	4305 of 4500	95.67 %	7819 of 7830	99.86 %
5	5	0.0005	30	121	0.0054	99.90 %	0.0038	99.92 %	4349 of 4500	96.64 %	7819 of 7830	99.86 %
6	0	0.001	30	21	0.0070	99.81 %	0.0027	99.92 %	4315 of 4500	95.89 %	7819 of 7830	99.86 %
7	5	0.001	20	121	0.0084	99.80 %	0.0035	99.92 %	4340 of 4500	96.44 %	7819 of 7830	99.86 %
8	5	0.001	30	121	0.0064	99.86 %	0.0039	99.94 %	4289 of 4500	95.31 %	7819 of 7830	99.86 %
9	10	0.001	30	221	0.0075	99.82 %	0.0073	99.90 %	4296 of 4500	95.47 %	7819 of 7830	99.86 %
10	0	0.0005	50	21	0.0058	99.85 %	0.0050	99.89 %	4320 of 4500	96.00 %	7818 of 7830	99.85 %

Tabla 4.1: Mejores modelos según la tasa de acierto en las predicciones

4.2. MÉTRICAS

A continuación se realiza el análisis de los modelos 1, 5 y 10 de la Tabla 4.1.

- **Modelo número 1**: Para tener una rápida aproximación al desempeño de este modelo, se analiza la matriz de confusión[2] mostrada en la Fig. 4.1. Cada columna representa el número de predicciones realizadas por este modelo para cada uno de los 9 gestos, mientras que las filas representan el gesto verdadero. Para estas predicciones, se utiliza el conjunto de muestras de prueba del dataset propio compuesto por 7 830 muestras (870 para cada gesto). Vale aclarar que este conjunto de prueba es independiente del conjunto utilizado para el entrenamiento/validación.

  Desglosando un poco la información de esta matriz de confusión, se pueden observar, en la Tabla 4.2, las predicciones incorrectas. Allí se visualiza en la primera columna el gesto predicho por el modelo, en la segunda columna el gesto verdadero y en la tercera columna el número de veces que hubo confusión.

  En la Tabla 4.3 se presentan las métricas[3] que significan lo siguiente:

  - **Precision**: Proporciona información sobre los falsos positivos, como se muestra en la Ec. 4.1. Es la relación entre los casos positivos bien clasificados y el número total de predicciones realizadas.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4.1}$$

    donde:

    TP es la cantidad de verdaderos positivos (TP: True Positives).
    FP es la cantidad de falsos positivos (FP: False Positives).

[2]Una matriz de confusión permite visualizar los resultados de las predicciones obtenidas por un clasificador.

[3]Las métricas son: Precision (Precisión), Recall (Exhaustividad), F1-score (Valor-F) y Accuracy (Exactitud). Se utilizarán sus nombres en inglés.

- **Recall**: Indica la proporción de clases positivas que el modelo ha podido predecir correctamente. Para ejemplificar, si la proporción es demasiado baja, significa que el modelo ha fallado demasiados positivos. Siendo FN el número de falsos negativos, recall se define en la Ec. 4.2.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4.2}$$

- **F1-score**: Combina precision y recall en un sólo valor y permite comparar el rendimiento entre varios modelos. El F1-score se define en la Ec. 4.3.

$$\text{F1-score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{4.3}$$

- **Support**: Cantidad de predicciones realizadas para cada clase.
- **Accuracy**: Mide la proporción de casos en los que el modelo ha tenido éxito, considerando todas las clases. Es decir, es la tasa de aciertos.

De esta información se puede mencionar que el modelo tiene precision del 100 % para los gestos *W*, *9* y *A*, lo que significa que en ninguna de las predicciones realizadas ha resultado el gesto *W*, *9* o *A* cuando no lo eran. Esto se puede comprobar en la columna "Predicción" de la Tabla 4.2 en la que no aparecen los gestos *W*, *9* y *A*.

Por otro lado, en la columna "Verdadera" de la Tabla 4.2 no aparecen los gestos *S*, *4*, *5* y *L*, lo que significa que tienen un 100 % de recall. Esto significa que todas las predicciones realizadas para los gestos *S*, *4*, *5* y *L* han sido acertadas sin tener predicciones incorrectas.

Predicción	Verdadera	Cantidad de preddicones incorrectas
L	1	1
L	A	1
5	9	1
V	W	1
4	W	2
1	V	2
S	A	2

Tabla 4.2: Cantidad de predicciones incorrectas del modelo 1

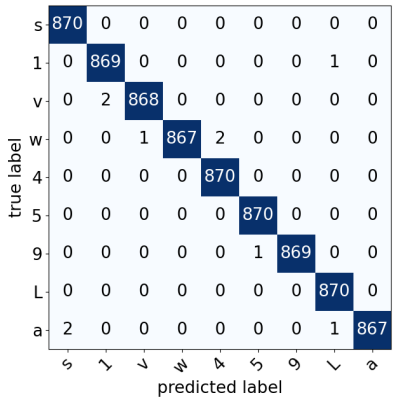


Figura 4.1: Matriz de confusión del modelo 1

gesture	precision	recall	f1-score	support
s	0.9977	1.0000	0.9989	870
1	0.9977	0.9989	0.9983	870
v	0.9988	0.9977	0.9983	870
w	1.0000	0.9966	0.9983	870
4	0.9977	1.0000	0.9989	870
5	0.9989	1.0000	0.9994	870
9	1.0000	0.9989	0.9994	870
L	0.9977	1.0000	0.9989	870
a	1.0000	0.9966	0.9983	870
accuracy = 0.9987			for 7830 predictions (870 each class)	

Tabla 4.3: Métricas de las predicciones realizadas con el modelo 1

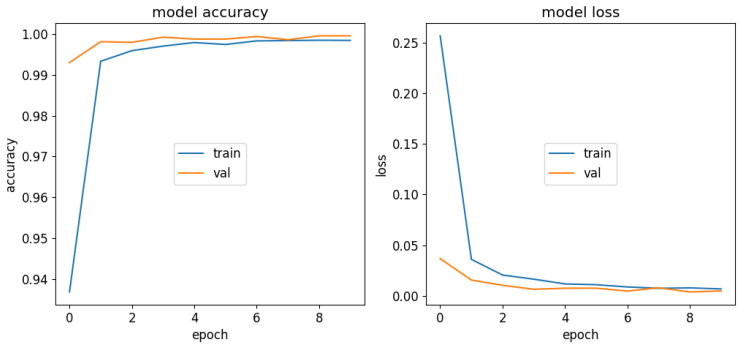


Figura 4.2: Accuracy y pérdida para el entrenamiento del modelo 1

En la Fig. 4.2, se registran las métricas de entrenamiento para cada época, incluyendo el accuracy y la pérdida para los conjuntos de entrenamiento y validación. Se observa un correcto aprendizaje de los parámetros de la red con el conjunto de

entrenamiento con el paso de las épocas y con el conjunto de validación se observa que a partir de la época número 6 no mejora el rendimiento de forma significativa. Cabe mencionar que en este modelo se utilizó una tasa de aprendizaje de 0.0005 para el optimizador Adam y que no se agregaron keypoints extras en los huesos de las manos.

- **Modelo número 5**: Para este modelo y de forma comparativa sólo se analizarán las métricas de la Tabla 4.4 y las gráficas de la Fig. 4.3. Se pueden observar algunas diferencias mínimas entre precision y recall con respecto al modelo número 1. Sin embargo, con la métrica f1-score se puede realizar una comparación e indicar que el modelo número 5 tiene más predicciones erróneas pero sigue estando muy cerca al rendimiento del modelo anterior. En cuanto a las métricas durante el proceso de aprendizaje de la red, se observa un comportamiento similar al modelo anterior, donde el rendimiento no mejora considerablemente a partir de la época número 10. Para este modelo se utilizó una tasa de aprendizaje de 0.0005 y se añadieron 5 keypoints adicionales a cada hueso, lo que hace un total de 121 keypoints para cada mano.

gesture	precision	recall	f1-score	support
s	0.9977	1.0000	0.9989	870
1	0.9977	1.0000	0.9989	870
v	0.9988	0.9977	0.9983	870
w	1.0000	0.9954	0.9977	870
4	0.9966	0.9989	0.9977	870
5	0.9977	1.0000	0.9989	870
9	1.0000	1.0000	1.0000	870
L	0.9989	0.9989	0.9989	870
a	1.0000	0.9966	0.9983	870
accuracy = 0.9986		for 7830 predictions (870 each class)		

Tabla 4.4: Métricas con modelo 5

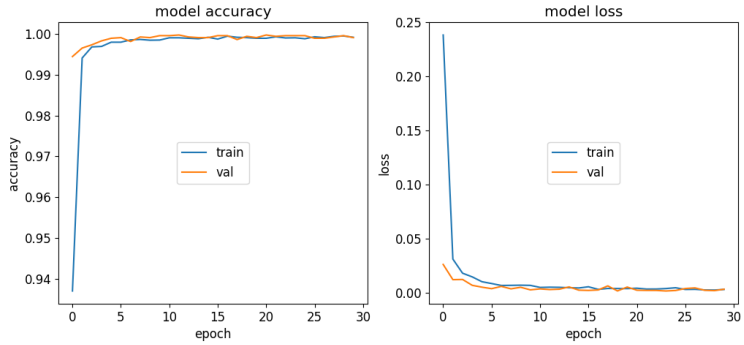


Figura 4.3: Accuracy y pérdida para el entrenamiento del modelo 5

- **Modelo número 10**: En la Tabla 4.5 se observa un desempeño similar a los modelos anteriores. No hay diferencias notables en las métricas durante el aprendizaje (Fig. 4.4).

gesture	precision	recall	f1-score	support
s	0.9977	0.9989	0.9983	870
1	0.9977	1.0000	0.9989	870
v	0.9977	0.9977	0.9977	870
w	1.0000	0.9954	0.9977	870
4	0.9966	1.0000	0.9983	870
5	0.9989	0.9989	0.9989	870
9	1.0000	1.0000	1.0000	870
L	0.9989	0.9989	0.9989	870
a	0.9988	0.9966	0.9977	870
accuracy = 0.9985		for 7830 predictions (870 each class)		

Tabla 4.5: Métricas con modelo 10

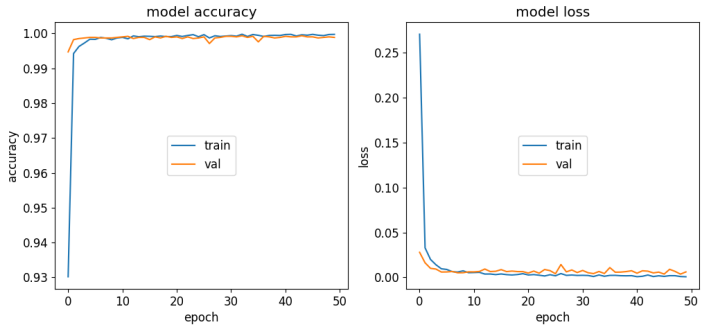


Figura 4.4: Accuracy y pérdida para el entrenamiento del modelo 10

Los resultados de la Tabla 4.1 muestran un alto rendimiento de la red propuesta. Se detecta que los keypoints extras añadidos en cada hueso tendrían poca importancia, lo que indica que estos keypoints extras no aportan información significativa.

Sería importante incluir otro tipo de información a los datos de entrada, como el ángulo de flexión en cada articulación y un número que identifique cada keypoint. Es decir, si se observa la Fig. 2.52 se puede ver que cada keypoint tiene un número que lo identifica y además en algunos keypoints está definido un ángulo de flexión (excepto en los keypoints de la muñeca y la punta de los dedos que no tienen un ángulo definido). De esta forma, los datos de entrada podrían definirse como (x, y, z, id, ángulo) de tal manera permita agregar esta información relevante.

4.3. Resultados comparativos

Se desea comparar la tasa de aciertos en las predicciones realizadas por el modelo propuesto en esta tesis sobre el conjunto de datos de prueba propio (7 830 muestras) y también con el conjunto de prueba externo (4 500 muestras). Cabe recordar que el conjunto propio es una extracción aleatoria del 20 % de las muestras del dataset completo y que el conjunto de pruebas externo es una colección de muestras de trabajos de terceros. Este conjunto de muestras externas se realizó con el fin de obtener datos diversos, ya que se extrajeron de imágenes tomadas en otros entornos y con gestos de otras personas.

Además de realizar las predicciones (con el modelo número 1 en la Tabla 4.1) en los dos conjuntos de prueba, también se utilizó el modelo PointNet [Seo and Joo, 2020] entrenado con el optimizador Adam con tasa de aprendizaje de 0,001 y 20 épocas, y también con un modelo creado a partir del propuesto en esta tesis, pero sin la etapa de transformación y normalización inicial. Se puede apreciar en estos resultados (Tabla 4.6) la importancia de la capa de transformación y normalización que se aplica inicialmente. Proporciona un aumento significativo de la tasa de aciertos cuando las predicciones se realizan con muestras muy variadas de diferentes fuentes de terceros.

Modelo	Accuracy (dataset propio)	Accuracy (dataset externo)
Modelo propio	7820 de 7830 99.87 %	4336 de 4500 96.36 %
PointNet	7660 de 7830 97.82 %	2155 de 4500 47.89 %
Modelo propio sin normalización	7760 de 7830 99.11 %	1371 de 4500 30.47 %

Tabla 4.6: Comparación de modelos

Comparación con ROC y AUC

La curva ROC (Receiver Operating Characteristic) es una medida de la calidad predictiva de un clasificador. La ROC compara y visualiza el compromiso entre la Tasa de Verdaderos Positivos (TVP = $\frac{\text{TP}}{\text{TP+FN}}$) y la Tasa de Falsos Positivos (TFP = $\frac{\text{FP}}{\text{FP+TN}}$). Las curvas ROC se utilizan normalmente en la clasificación binaria, pero una de las formas de abordarla es binarizando la salida (por clase). Una curva ROC muestra la tasa de verdaderos positivos en el eje Y y la tasa de falsos positivos en el eje X. Por tanto, la región ideal es la esquina superior izquierda del gráfico, donde los falsos positivos son cero y los verdaderos positivos son uno. Esto lleva a presentar el Área Bajo la Curva (AUC: Area Under the Curve), que es una métrica que relaciona los falsos positivos y los verdaderos positivos. Cuanto mayor sea el AUC, en general, mejor será el modelo.

La Fig. 4.5 presenta la curva ROC del modelo número 1 (de la Tabla 4.1) y muestra el alto porcentaje de éxito alcanzado en las predicciones. Los tres modelos predicen considerablemente bien con el conjunto de datos propio, como se muestra en la Tabla 4.6, y las curvas ROC son muy similares a la presentada en la Fig. 4.5.

Además se muestran las curvas ROC de los tres modelos realizando las predicciones con el conjunto de datos externo. Se puede observar que sólo el modelo propuesto en esta tesis con la capa de normalización y transformación se comporta con un rendimiento aceptable. Esto se muestra en la Fig. 4.6, 4.7 and 4.8.

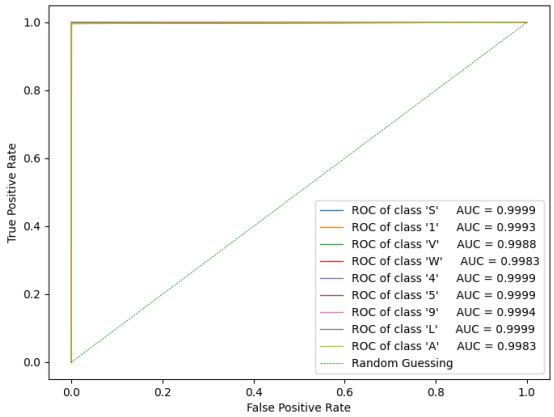


Figura 4.5: Curvas ROC y AUC del modelo propuesto con el dataset propio

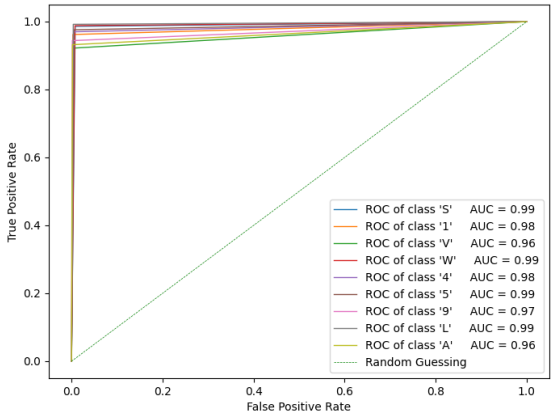


Figura 4.6: Curvas ROC y AUC del modelo propuesto con el dataset externo

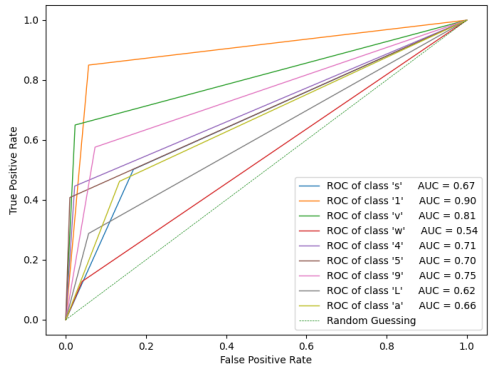


Figura 4.7: Curvas ROC y AUC del modelo PointNet con el dataset externo

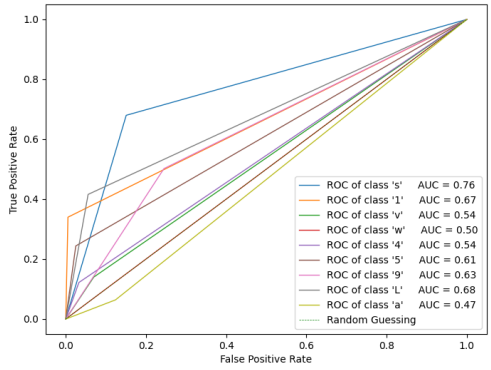


Figura 4.8: Curvas ROC y AUC del modelo propuesto sin normalización

4.4. Composición del modelo

Una vez culminado el entrenamiento con el dataset propio se obtiene el modelo entrenado, es decir, todos los parámetros entrenables de la red ya tienen sus valores ajustados. Con este modelo se pueden realizar las predicciones que permitirían alcanzar la tasa de aciertos que se presupone luego del análisis realizado con el conjunto de datos de prueba. La arquitectura de red propuesta en esta tesis tiene 12 617 parámetros entrenables, como muestra la tabla de la Fig. 3.20. Es una buena idea comprender el comportamiento de una red neuronal mediante la visualización de sus parámetros, ya sea durante el entrenamiento o cuando se realizan las predicciones con el modelo ya entrenado. Frecuentemente se intenta obtener una representación gráfica en cada capa de la red neuronal para observar de qué manera se van agrupando características de los datos de entrada y cómo se va propagando la información. Este tipo de representaciones es muy común cuando se utilizan redes neuronales convolucionales debido a que los datos de entrada a la red son imágenes y al realizar convoluciones con kernels, pueden dar como resultado otras imágenes a medida que se va propagando la información entre las capas. En esta tesis,

si bien los datos de entrada al sistema son imágenes, los datos que ingresan a la red perceptrón multicapa son los 21 keypoints, que no son otra cosa que 21 coordenadas (x, y, z). Al ser una red del tipo perceptrón multicapa, se hace difícil una visualización de la información propagándose entre las capas debido a la multidimensionalidad de los parámetros. Sin embargo, se pueden utilizar algunas estrategias para visualizar lo que sucede en cada capa.

A continuación se expondrá una representación visual de los parámetros entrenados de la red, para luego poder visualizar la propagación de la información al realizar las predicciones. Los parámetros ajustados durante el entrenamiento son los siguientes: valores de los kernels de las capas de convolución; desviación estándar y la media en los BatchNormalization; y los pesos de la capa Fully Connected y softmax.

En la Fig. 4.9 se muestra la primera etapa del sistema, que actúa para la extracción de los 21 keypoints utilizando MediaPipe, para luego realizar las transformaciones de traslación, rotación y escalado que normalizan esos keypoints para dejarlos preparados para el ingreso a la segunda etapa del sistema, que es la red perceptrón multicapa. En esta figura y en las siguientes, los sectores que se visualizan en tonalidad marrón están destinados a información que se irá propagando por las distintas capas para realizar las predicciones. Mientras se encuentren en tonalidad marrón es porque aún no contiene información.

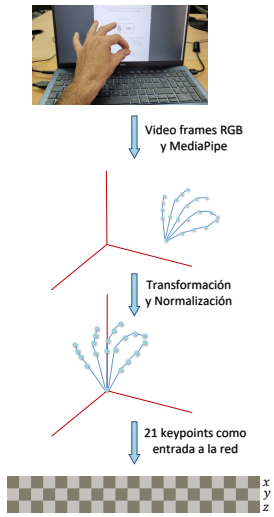


Figura 4.9: Obtención de keypoints y transformaciones para normalización

En la Fig. 4.10 se incorporan los 32 kernels de la primera capa de convolución de la red, con sus valores aprendidos durante el entrenamiento. Con estos kernels con sus valores (en tonalidad roja) se realiza la convolución unidimensional de cada uno de estos 32 kernels con los 21 keypoints para generar una matriz de valores de 21×32, que son normalizados con el BatchNormalization para luego aplicar la función de activación ReLU. En esta figura, los valores aprendidos de los 32 kernels son mostrados en colores rojos con distintas intensidades, desde la ausencia de color para los valores menores hasta el color con la máxima intensidad para los valores mayores.

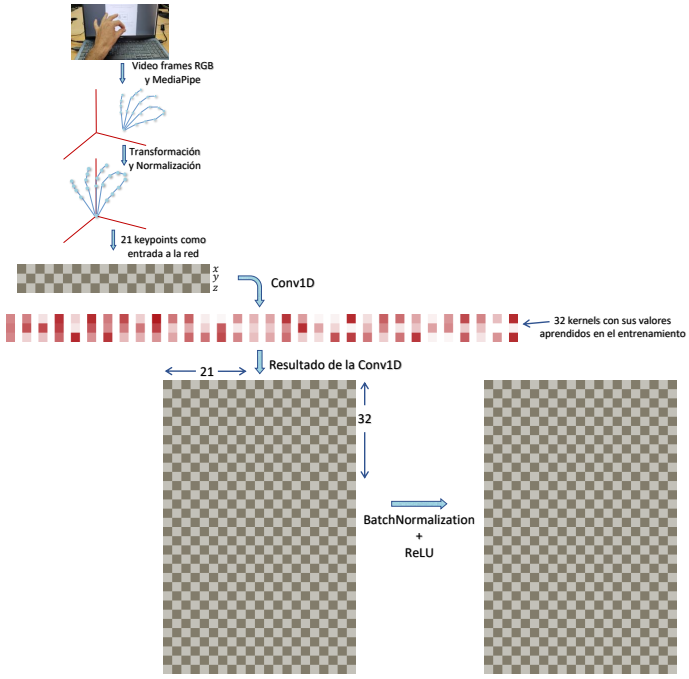


Figura 4.10: Parámetros de la primera capa

La Fig. 4.11 muestra la segunda capa de convolución unidimensional que posee 64 kernels con 32 parámetros entrenables cada uno, lo que entrega, luego de convolucionar, una salida de 21×64 valores. Luego se normalizan con BatchNormalization y se aplica la función ReLU. Después se realiza un pooling que por cada fila de valores se conserva el mayor, es decir, un max pooling.

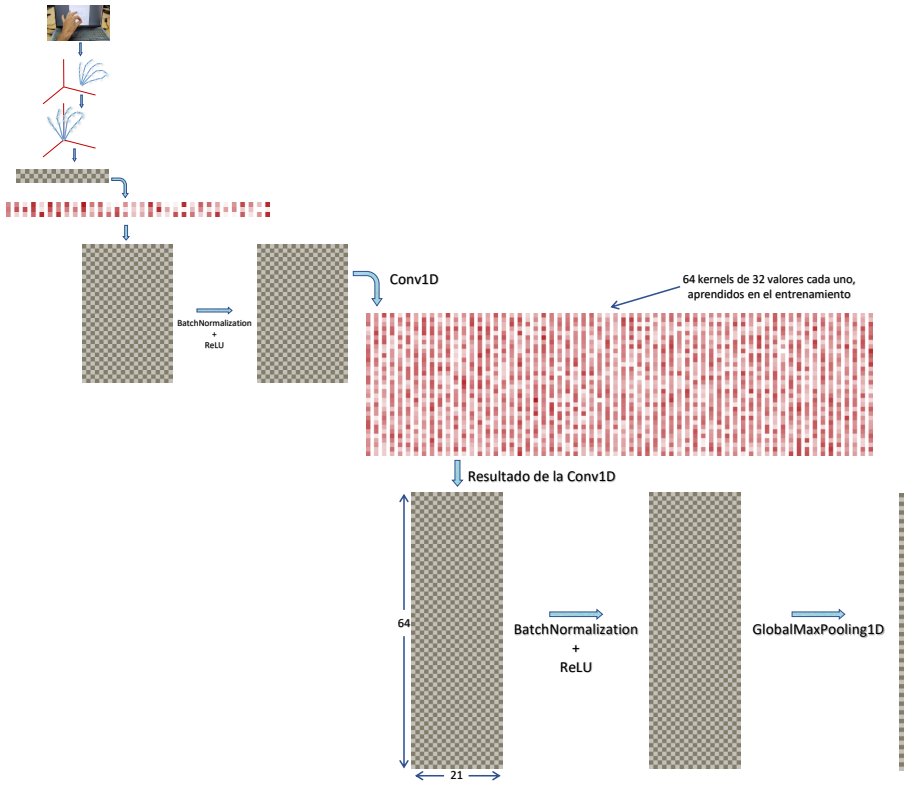


Figura 4.11: Parámetros de la segunda capa y Max Pooling

En la Fig. 4.12 se muestra la última etapa de la red perceptrón multicapa propuesta en esta tesis. Allí se observan, en tonalidades verdes y azules, los pesos aprendidos en las últimas capas que son Fully Connected y entregan la predicción en su salida softmax. En esta gráfica se enumeran con flechas rojas, aquellos sectores de la red en donde se almacenan los resultados de los cálculos que se realizan en cada etapa a medida que en la entrada se presentan los 21 keypoints con el objetivo de realizar una predicción. La información de entrada se va propagando con las operaciones de la red con sus parámetros entrenados y presentará en la salida softmax la predicción realizada.

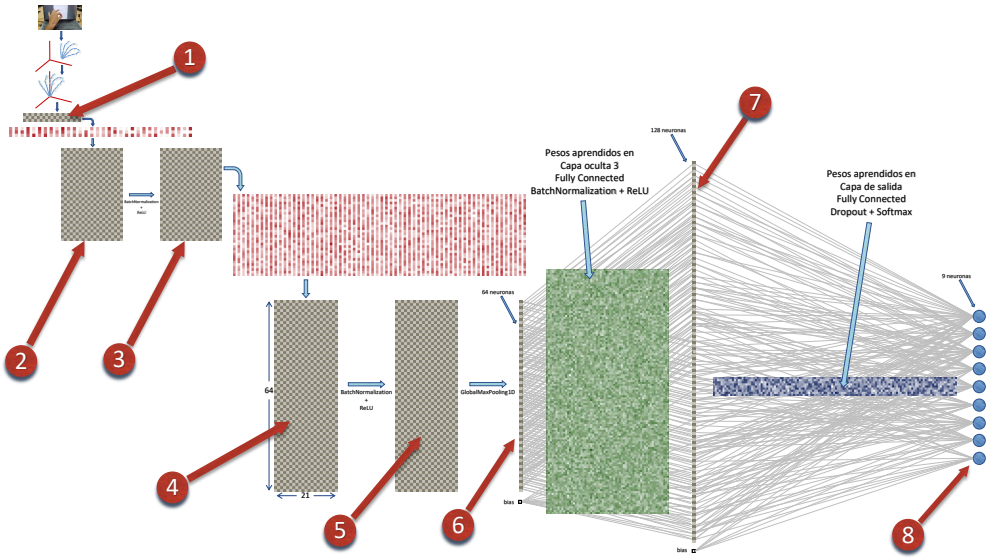


Figura 4.12: Sistema completo con parámetros entrenados

Hasta aquí se presentaron los parámetros entrenados de la red, y a continuación se mostrará la información propagándose por la red cuando se desea realizar una predicción. En la Fig. 4.13 se muestra, en tonalidades azules, la información propagándose por la red cuando en la entrada se colocan los keypoints de un gesto de mano en puño. En esta gráfica se encuentra toda la información que se va transformando al superar cada capa entrenada. Algunas cuestiones para resaltar es la visualización evidente de la función de activación ReLU en donde se observa que coloca en cero a los valores negativos, es por ello que la aplicación de las tres funciones ReLU, que es una función del tipo max(0, entrada), lleva a que muchos valores se hagan cero.

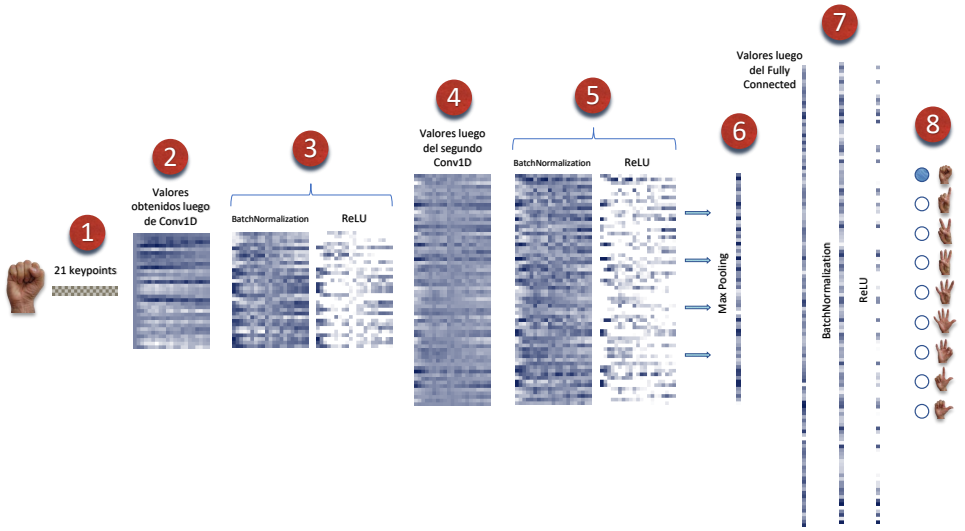


Figura 4.13: Propagación de información en la red para predicción del gesto S

Sólo a modo comparativo se muestra en la Fig. 4.14 la propagación de la información cuando se ingresan los keypoints de otro gesto. Allí no se logra identificar un patrón visual como sí se puede observar en otros trabajos que utilizan redes neuronales convolucionales que, a medida que se convolucionan imágenes con kernels y se avanza en las capas de red, se van agrupando patrones visuales. Algunos trabajos, como [Wang et al., 2021] y [Mostafa et al., 2021] están atentos a interpretar el proceso de entrenamiento y predicción, extrayendo la información de las capas a medida que se realiza el entrenamiento. Se podría pensar que es propio de las redes neuronales convolucionales obtener representaciones visuales de la información que se propaga por las capas, ya que los kernels o filtros aplicados generan resultados que mantienen esas características visuales. Para el caso de redes del tipo perceptrón multicapa se deben realizar proyecciones hacia un plano 2D o un espacio 3D para poder visualizarlo a ojo humano, como se hace en el trabajo [Cantareira and Paulovich, 2020]. Los autores realizan una reducción de la dimensionalidad de los datos para poder presentarlos en una gráfica y mientras la

información se va propagando por la red, se van agrupando características que indican de qué manera se va conformando una predicción.

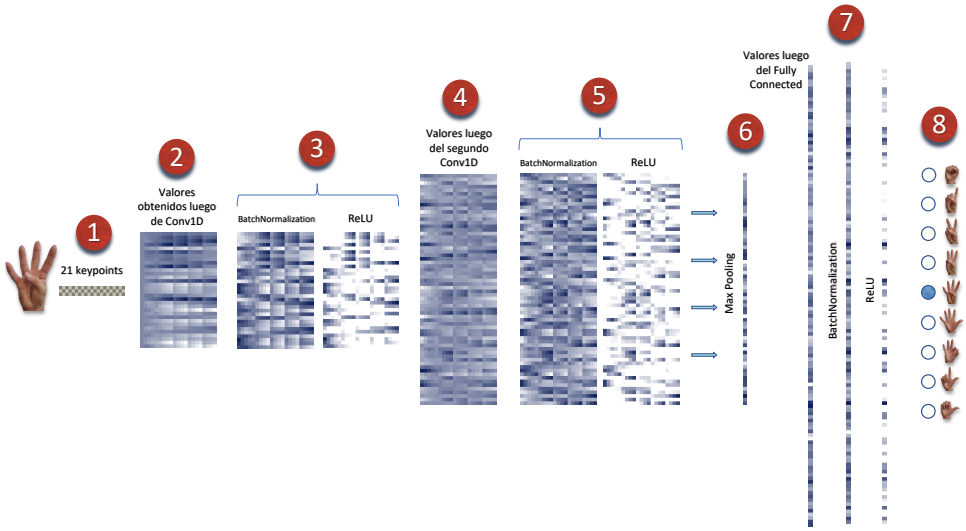


Figura 4.14: Propagación de información en la red para predicción del gesto 4

4.5. Conclusiones sobre el modelo

En este trabajo, se presenta una nueva arquitectura de red para el reconocimiento de gestos de la mano utilizando una nube de puntos. El estudio se centró en la nube de puntos de baja densidad obtenida mediante una cámara RGB estándar. La nueva red (inspirada en la arquitectura PointNet) fue entrenada con keypoints de la mano y gracias a una arquitectura simple con pocas capas ocultas es posible trabajar directamente con CPU.

Los resultados muestran una tasa de aciertos del 99,87 % en el conjunto propio de datos de gestos de la mano. Es interesante ampliar este estudio incluyendo nuevos gestos para tener una mayor variedad de opciones para el control de dispositivos, y también experimentar con los usuarios finales para detectar aquellos gestos que son más apropiados para realizar determinados comandos de control.

Es importante destacar que la capa de transformación y normalización permite mantener el buen rendimiento en las predicciones utilizando conjuntos de datos de terceros que contienen una gran variedad de usuarios y espacios físicos donde se tomaron las muestras.

Capítulo 5

Implementación del modelo en una aplicación

Capítulo 5

Implementación del modelo en una aplicación

Contenidos

En este capítulo se presenta el desarrollo de un controlador que permite tomar el mando del teclado y mouse de una computadora con los gestos de las manos. Los 9 gestos seleccionados en este trabajo para crear el dataset y posteriormente generar el modelo de clasificación fueron aquellos de uso más común cuando se intenta tener una comunicación no verbal entre personas y los más comunes a la hora de imaginar un control remoto de dispositivos. Esto último hace referencia a que si se pudiera imaginar un mando a distancia con las manos, las personas haríamos un gesto con dedo índice extendido para indicar que se está señalando algo, o con el índice haciendo un movimiento hacia delante como si se hiciera click sobre el aire, la mano en puño para indicar que se agarra algo, y abrir la mano para soltar, el pulgar arriba para indicar que es correcto, entre tantos otros gestos y movimientos. Justamente en esto se reflexionó para decidir cuáles serían los gestos, posturas, secuencia de gestos y movimientos adecuados para definir las distintas acciones de mando. Con el fin de limitar la gran cantidad de gestos posibles que se podrían incluir, se descartó el uso de las dos manos del usuario. Vale aclarar también que el modelo que fue entrenado identifica los 9 gestos, pero además se tienen las coordenadas tridimensionales de cada uno de los 21 keypoints de la mano. Entonces, la implementación utiliza toda esta información para diseñar este sistema de control.

A continuación se muestran las principales características que tiene este sistema de mando, comenzando por el nombre: **Hand Controller**. Y una pieza de identidad visual (Fig. 5.1). Está desarrollado en lenguaje Python con la utilización de las siguientes bibliotecas principales: Keras, OpenCV y Qt[1]. Disponible para su instalación en sistemas operativos Windows 10, Windows 11 y GNU/Linux en sus distribuciones Ubuntu 18.04 o superiores.

Figura 5.1: Logo de Hand Controller

[1]Qt (Quasar Technologies) es una biblioteca que facilita la creación de interfaces gráficas de usuario.

5.1. Apariencia

Hand Controller es un controlador que, a través de una cámara RGB, procesa las imágenes identificando ciertos gestos de las manos y, en base a ello, genera eventos de teclado y mouse para actuar sobre la computadora. Es decir, se podrá controlar el sistema operativo como si se estuviera utilizando el teclado y mouse. Con estas características de Hand Controller, es oportuno mencionar que se ejecuta en segundo plano, sin interferir en las aplicaciones que se puedan estar utilizando. Dos aspectos en los que se interpone son: la utilización de una cámara RGB, dejando sin la posibilidad de utilizar dicha cámara para otra función, y en la visualización de una miniatura de las imágenes capturadas por la cámara en un sector de la pantalla (Fig. 5.2), con el fin de que el usuario pueda observar sus movimientos en tiempo real.

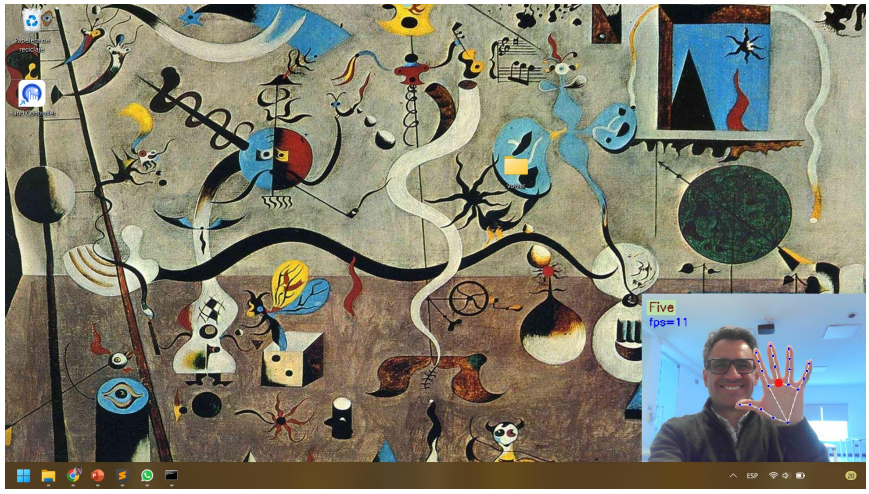


Figura 5.2: Ubicación de la miniatura

Existen muchos aspectos configurables y personalizables que se comentarán en este capítulo a medida que se avance en la descripción de las características de Hand Controller. Por ejemplo, la cámara RGB que utiliza se selecciona a gusto del usuario si tiene más de una cámara o toma una por defecto, ya sea la cámara integrada en una computadora portátil o una cámara web conectada por USB. Las imágenes que entrega la cámara pueden tener diversos tamaños dependiendo el modelo de cámara y la configuración en el sistema operativo. Hand Controller definirá su propio tamaño de captura de imágenes y es 640×480 píxeles. Por otro lado, la visualización de la miniatura también se puede personalizar, eligiendo su posición en pantalla (por defecto, abajo a la derecha), tamaño (por defecto, 400×300) y transparencia (por defecto, 0.82). Además se puede configurar la visualización dentro de la miniatura alguna información como: fps, nombre del gesto detectado, esqueleto de la mano, keypoints y algunas líneas, recuadros y puntos que facilitan al usuario conocer qué acción está realizando. Además se desarrolló un teclado virtual que se muestra en la Fig. 5.3 y es configurable su tamaño y transparencia.

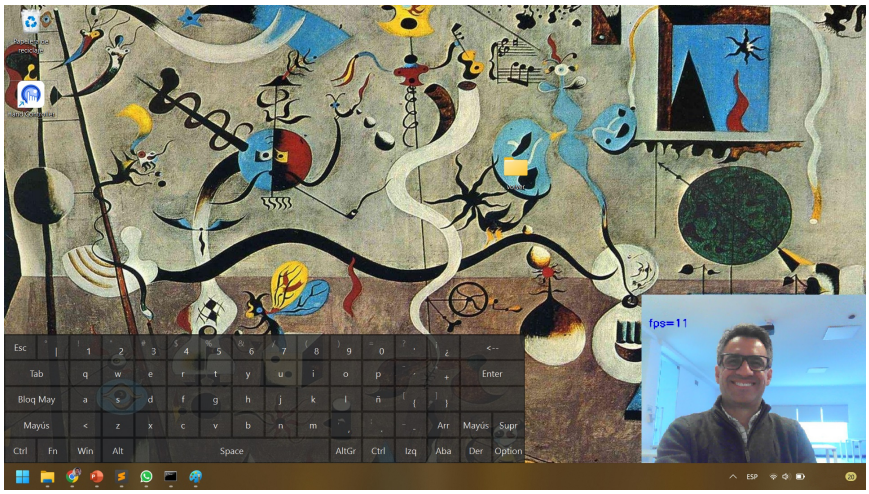


Figura 5.3: Teclado virtual

5.2. Gestos para la interacción

En esta sección se describen los gestos para realizar el control. Antes de ello es necesario mencionar que, con el motivo de uniformizar el desempeño de Hand Controller en computadoras que tengan distintas capacidades de procesamiento y tiempos de cómputo, se realiza un control de los fotogramas por segundo que se procesan. Este control se realiza mediante el propio Hand Controller y está estabilizado en un aproximado de 11 fps. Esta elección es adecuada para que el sistema se pueda utilizar en la mayoría de las computadoras con las prestaciones que hoy en día frecuentan. Estabilizar los fotogramas por segundo permite que la velocidad de respuesta sea independiente de la computadora que se esté usando.

De todo el abanico disponible de gestos que el modelo detecta y la información de las coordenadas de cada keypoint para realizar el control, se seleccionan los siguientes para cada acción:

- **Desplazamiento del mouse**: Keypoint[2] número 9 con la mano abierta (nombre de gesto: 5)[3]. Con la mano abierta se controla el desplazamiento del puntero del mouse por toda la pantalla, y toma los valores (x, y) del keypoint número 9, que es el comienzo del dedo mayor. Cuando se está realizando este control, se dibuja un punto rojo en dicho keypoint para indicarle al usuario que está controlando el mouse. Debido a que es común que el rostro del usuario se encuentre enfrentado a la pantalla de la computadora y como también es frecuente que la cámara se encuentre justo por encima de la pantalla, puede suceder que

[2]Ver la Fig. 2.52 para tener presente el número de cada keypoint.
[3]Ver Fig. 3.12 para los nombres asignados a cada gesto.

la mano del usuario cuando está desplazando el mouse, obstruya su propia visión con la pantalla. Para evitar este inconveniente, se seleccionó una región delimitada por un rectángulo verde (ver Fig. 5.4) que indica la región en donde el keypoint 9 tiene acción de desplazamiento. Dicho de otra manera, cuando el keypoint 9 con la mano abierta se mueve por la totalidad de la región definida por el rectángulo verde, abarcará el desplazamiento del mouse por la totalidad de la pantalla. El rectángulo verde es configurable en su ubicación y por defecto está ubicado hacia la derecha, pensado para usuarios que usen su mano derecha. Usuarios que usen su mano izquierda deberán configurar no sólo el rectángulo verde en la izquierda sino también la ubicación de la miniatura de la cámara en la izquierda.

Figura 5.4: Desplazamiento del mouse y rectángulo verde

- **Scroll vertical**: Keypoint número 9 con la mano en puño (gesto S) y desplazamiento vertical. Aquí entra una característica que es la detección de secuencias de gestos, es decir, se almacena un historial de gestos distintos que el usuario realiza. Para realizar el scroll vertical se requiere poner la mano en puño y venir desde el gesto con mano abierta. En el mismo instante que se realiza la secuencia mano abierta hacia puño (secuencia 5/S) se almacena la posición del keypoint 9 y se calcula el desplazamiento hacia abajo o hacia arriba de ese mismo keypoint en los fotogramas siguientes. Mientras mayor es la distancia vertical entre la posición inicial del keypoint 9 y su posición en los siguientes fotogramas, más pronunciado será el scroll. Ver Fig. 5.5.

- **Teclas izquierda y derecha, o scroll horizontal**: Keypoint número 9 con la mano en puño y desplazamiento horizontal (Fig. 5.6). Algo importante para mencionar es que existen algunos gestos que resultan naturales para ser usados en distintas acciones, por ejemplo, el puño cerrado con desplazamiento hacia la derecha o hacia la izquierda podría ser utilizado tanto para un scroll horizontal o para presionar las techas izquierda y derecha. En Hand Controller se debe configurar previamente cuál de estas acciones es la que se desea con ese gesto. Siguiendo el mismo mecanismo que el utilizado para el scroll vertical, se realiza el scroll hori-

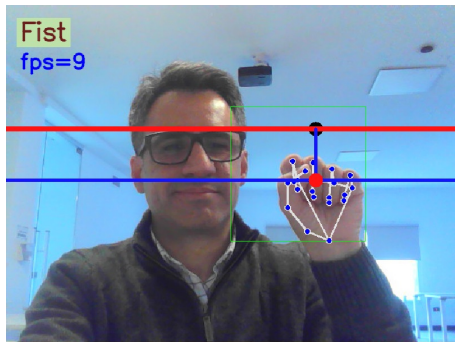


Figura 5.5: Scroll vertical

zontal o el pulsado de las teclas izquierda o derecha, dependiendo cómo se haya configurado. Es de utilidad las teclas izquierda y derecha para realizar el avance y retroceso en una exposición de diapositivas.

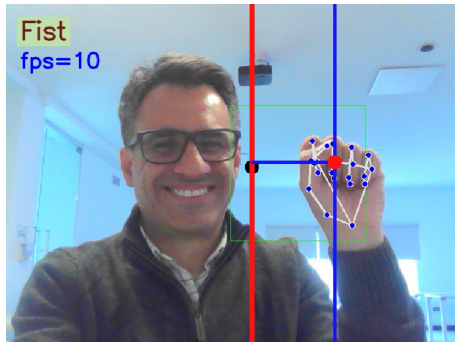


Figura 5.6: Teclas izquierda y derecha

- **Drag and Drop**: Mano abierta para desplazar el mouse, cambiar a puño para agarrar (equivalente a pulsar el click izquierdo del mouse y mantenerlo presionado sin soltar), desplazar con el puño y abrir la mano para soltar. Es natural para un usuario que esta acción de drag and drop se realice con el puño, pero esto interfiere con el gesto del puño para realizar scroll. Por este motivo, Hand Controller está configurado para poder funcionar con Drag and Drop, o con Scroll. Debe ser configurado previamente o se puede activar o desactivar uno o el otro con la secuencia de gestos L/A/L/A.

- **Click izquierdo (Opción número 1)**: Mano con el índice extendido (gesto 1) y desplazamiento del keypoint número 8 hacia delante y luego atrás. La elección del gesto para realizar un click no fue trivial, por ello se diseñaron dos opciones que se pueden configurar en Hand Controller. Esta primera opción se realiza co-

menzando por la secuencia 5/1 (mano abierta y mano con sólo el índice extendido) y a partir de este momento se considera la punta del índice (keypoint 8) para que el usuario lo acerque y lo aleje en dirección a la cámara. Es un gesto muy similar a pulsar un botón imaginario que está suspendido en el aire. Aquí entran en juego distintos parámetros que fueron ajustados con valores por defecto y además pueden ser configurados dependiendo el caso. Estos parámetros son: distancia recorrida que debe hacer el índice para acercarse y alejarse de la cámara y tiempo en el que se deben hacer estos movimientos. Para realizar estos cálculos es importante la decisión de estabilizar la cantidad de fotogramas por segundo. Una cuestión no menor es que al cambiar de mano abierta a mano con el índice extendido, el puntero del mouse tiene un leve cambio de posición sólo por el hecho que la mano está en el aire sin una estabilidad rigurosa. Por ello se aplicó un cambio y se agregó un gesto que permite un desplazamiento mucho más preciso y lento sobre un sector reducido. El gesto elegido para esto es el gesto L. Entonces, la secuencia para realizar un click es: desplazar el mouse por todos los sectores de la pantalla con la mano abierta, luego cambiar a mano en L para movimientos precisos y de poca trayectoria, luego mano con índice extendido para pulsar ese botón imaginario.

- **Click izquierdo (Opción número 2)**: Mano con gesto de L y cambio a mano con el índice extendido (gesto 1). Una variante para realizar el click es desplazar el mouse por todos los sectores de la pantalla con la mano abierta, luego cambiar a mano en L para movimientos precisos y de poca trayectoria para luego cambiar a índice extendido para realizar el click. El paso de mano en L hacia mano con índice extendido puede comprenderse como la secuencia de colocar la mano en L y simplemente cerrar el pulgar.

- **Click derecho**: Mano con gesto V y desplazamiento del keypoint número 8 hacia delante y luego atrás. Para realizar el click derecho del mouse se definió una secuencia con las mismas características que el click izquierdo opción 1, pero en lugar de utilizar el gesto 1, se utiliza el gesto V.

- **Abrir teclado virtual**: Teniendo disponible un historial de los gestos que el usuario realiza durante el uso de Hand Controller, entonces pueden ser configuradas distintas acciones sólo con las secuencias de gestos, sin preocuparse de las coordenadas de los keypoints. Abrir y cerrar el teclado virtual propio de Hand Controller se realiza con la secuencia 9/A (ver Fig. 3.12 para los nombres de los gestos).

- **Cerrar Hand Controller**: Secuencia de gestos W/V/1/S.

5.3. Desplazamiento suavizado

Cuando el usuario tiene la mano abierta toma control sobre el desplazamiento del puntero del mouse, el cual será desplazado en función del desplazamiento del keypoint 9, que corresponde con el inicio del dedo mayor. Como se mencionó anteriormente, en la miniatura de la cámara que se visualiza en la pantalla se dibuja un rectángulo verde para que el usuario tenga la referencia de los márgenes habilitados para recorrer toda la pantalla con el puntero del mouse (ver Fig. 5.4). El rango de este rectángulo verde se adapta a la resolución de la pantalla para que el recorrido del mouse alcance cualquier sector de la pantalla. Como la cantidad de fotogramas por segundo son aproximadamente 11 fps, esto quiere decir que se procesa una imagen cada 91 milisegundos. Si se actualiza la posición del puntero del mouse cada 91 ms, puede suceder que el puntero realice saltos grandes en su desplazamiento. Para que el recorrido del puntero tenga una estela y no sean tan bruscos sus cambios de posición, se implementó un historial con las posiciones antiguas del keypoint 9 para que la actualización de la posición del puntero sea paulatina. Para ello de define un factor λ que representa el porcentaje de la distancia entre las dos últimas posiciones del puntero que se desplaza (ver Ec. 5.1). El valor por defecto para λ es de $0,15$.

$$\begin{aligned} x_{\text{nueva}} &= x_{\text{detectada}} + \lambda \left(x_{\text{anterior}} - x_{\text{detectada}} \right) \\ y_{\text{nueva}} &= y_{\text{detectada}} + \lambda \left(y_{\text{anterior}} - y_{\text{detectada}} \right) \end{aligned} \tag{5.1}$$

donde:

$(x_{\text{nueva}},\ y_{\text{nueva}})$ es la posición del puntero a la cual se actualizará

$(x_{\text{detectada}},\ y_{\text{detectada}})$ es la posición detectada del keypoint 9 en ese momento

$(x_{\text{anterior}},\ y_{\text{anterior}})$ es la posición del puntero en el momento anterior

5.4. Integración en ambientes inteligentes

Los ambientes inteligentes son aquellos entornos en donde la tecnología colabora con lo cotidiano y esos sistemas tienden a ser cada vez más imperceptibles. Cuando se dice que la tecnología colabora, se encuentra implícita una interacción entre las personas y los sistemas. Se encuentran cada vez más integradas al entorno aquellas tecnologías de monitoreo e interacción, tal como las cámaras, sensores de todo tipo, reconocimiento del habla, identificación de las personas, sistemas embebidos, computación ubicua, servicios en la nube, hardware personal como gafas, teléfonos y relojes inteligentes y una vasta cantidad de software y hardware distribuido para funcionar como un soporte cada vez más primordial para los humanos. El trabajo de esta tesis ofrece una opción más para ser integrado en ambientes inteligentes.

Acotando las posibilidades en busca de ser concretos en la aplicación, se pueden identificar algunos ambientes en donde el uso de Hand Controller puede adaptarse a las necesidades. Entre ellos se puede mencionar un campus universitario o un edificio

público en los cuales es frecuente encontrar pantallas que muestran información a las personas que concurren y se desplazan en estos espacios. En las universidades se publica información sobre los eventos del día, ubicación de las oficinas, dictado de cursos actuales, ofertas de postgrado, etc. y para el caso de edificios públicos, información sobre trámites, horarios de atención de las distintas oficinas, entre otros. Si en estos puntos de información se desea ofrecer al usuario la posibilidad de interactuar con el sistema y que pueda controlar la visualización de la información y, además, no se quiere colocar un teclado y mouse a disposición, entonces bajo esas condiciones, Hand Controller puede ser una opción interesante. Alcanza con colocar una cámara web y configurar Hand Controller para adaptarlo a las necesidades particulares en donde se coloque esta pantalla de información.

Existen trabajos relacionados, como [Patil et al., 2022], donde los autores presentan una interfaz gráfica de usuario controlada con los gestos de las manos con opciones de aplicación en cajeros automáticos, para check-in en aeropuertos, para realizar pedidos en restaurantes o comprar boletos para salas de cine. Los autores remarcan la preferencia de los usuarios en utilizar interfaces que no requieran el contacto con los dispositivos para una interacción más higiénica. En tal sentido, se puede encontrar una gama de quioscos interactivos [Ultraleap, 2022] que ofrecen alternativas sin contacto, permitiendo una interacción higiénica en espacios públicos. En la Figura 5.7 se visualizan algunas aplicaciones.

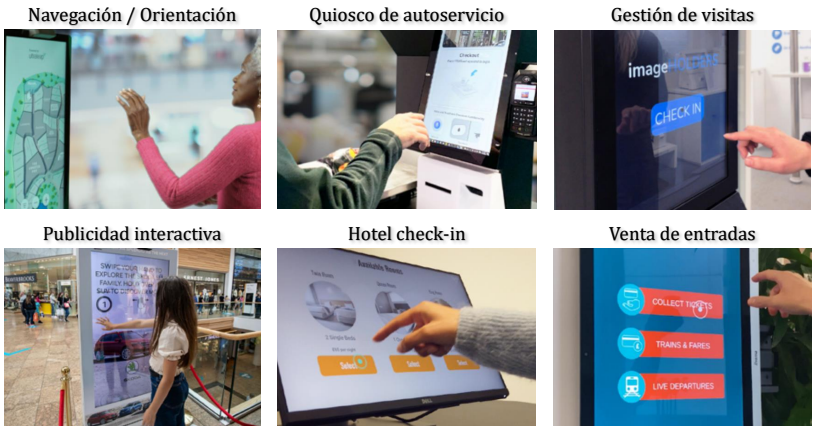


Figura 5.7: Soluciones ofrecidas por Ultraleap
Fuente: Ultraleap - https://www.ultraleap.com/success-kiosk-product-catalogue/

Otro espacio muy interesante de aplicación es en las plantas industriales, en donde frecuentan máquinas de gran porte y herramientas para la elaboración de productos. En las industrias, con el fin de aumentar la productividad, continuamente se adopta tecnología y muchas veces con la incorporación de pantallas para visualizar información de los procesos. En estos ambientes, donde muchas veces está presente la mano de obra que incluye un esfuerzo físico alto de los operarios, el uso de periféricos como el mouse, teclado o pantalla táctil puede ser una opción no recomendable, sobre todo para la

durabilidad de tales periféricos. Por lo que en este contexto, puede ser útil una opción alternativa de control como Hand Controller. También es interesante el control con los gestos de las manos en ambientes asépticos como los quirófanos, en donde mientras menos materiales, herramientas e instrumental se tenga que tocar y manipular, es más recomendable.

Una cámara procesando las imágenes de un lugar por donde pueden circular muchas personas, puede ser un problema a la hora de detectar cuándo algún usuario desea interactuar con el sistema y cuándo no. Algunas reglas simples pueden ser de ayuda, como ser la longitud del esqueleto de la mano como indicativo de cercanía del usuario hacia el sistema y con ello concluir que el usuario quiere interactuar. Sin embargo, esto es insuficiente, se requieren más indicativos. Por este motivo, se implementa en Hand Controller la detección de rostros para determinar si el usuario está enfrentado con la pantalla como señal que desea interactuar, también el tamaño del rostro detectado para tener una estimación de la distancia a la que se ubica, y detectar los gestos de las manos cuando el rostro esté presente y se encuentre a una distancia adecuada. Al cumplirse estas reglas, se puede suponer que el usuario se encuentra al frente de la pantalla con intenciones de interactuar. Como característica opcional orientada al uso eficiente de la energía, si el usuario no está cerca de la pantalla y no se encuentra de frente, entonces que la pantalla permanezca apagada.

Para la detección del rostro se integra MediaPipe Face Detection[4] a Hand Controller y se agregan opciones de configuración para definir un rango de distancia para que un usuario sea considerado como usuario que desea interactuar, una opción para habilitar el Face Detection para determinar si el usuario está de frente y dirigiendo su mirada hacia la pantalla y una opción para habilitar todo este mecanismo de detección de la intención de interacción.

[4]MediaPipe Face Detection - `https://google.github.io/mediapipe/solutions/face_detection.html`

Capítulo 6

Conclusiones y trabajos futuros

Capítulo 6

Conclusiones y trabajos futuros

Contenidos

En este último capítulo se presenta una síntesis de los aspectos más importantes de esta tesis para finalmente realizar las conclusiones generales, destacar las contribuciones originales y plantear los trabajos futuros.

6.1. Síntesis

Durante el progreso de este trabajo se tomaron algunas decisiones con el objetivo de mantener las siguientes dos características de la propuesta: "Utilización de cámaras RGB de bajo coste" y "Desempeño en tiempo real".

Entre las decisiones más destacadas se encuentra la elección de técnicas de detección basadas en redes neuronales artificiales que, en los últimos años, ha tenido un extraordinario avance en cuanto a la tasa de aciertos y a la reducción en los tiempos de procesamiento.

A continuación se hace una síntesis del desarrollo de esta tesis:

- *Sustracción de fondo y análisis geométrico de las manos*: En la primera etapa se exploraron e implementaron técnicas de detección y segmentación basadas en el análisis del color de los píxeles con el fin de aislar la mano, detectando su perímetro. Esto permite sustraer el fondo de la imagen para posteriormente realizar un análisis geométrico identificando las comisuras y la punta de los dedos. Este método de detección requiere que el fondo en la imagen tenga un color uniforme distinto al color de la piel y que la mano se visualice en una posición frontal a la cámara, ya que de otra manera las comisuras podrían ocluirse y no ser identificadas correctamente. Además de evaluar estas técnicas que estaban siendo usadas en el estado del arte, se adquirieron habilidades y conocimientos para la rápida construcción de prototipos.

- *Segmentación de las manos con sensor D-RGB*: En una siguiente etapa se utilizó el sensor Kinect para aislar las manos del fondo y ocuparse del análisis geométrico. Con este método no se requiere que el fondo tenga un color uniforme ya que las manos logran aislarse porque se encuentran cercanas al sensor. Igualmente este método fue descartado porque se buscaba utilizar cámaras RGB de bajo coste y de uso masivo.

- *Redes neuronales convolucionales para la detección del esqueleto de las manos*: Durante los años de desarrollo de esta tesis se fueron produciendo interesantes avances en el estado del arte respecto a la detección de los gestos de las manos, principalmente enfocado a la lengua de señas. El autor de esta tesis realizó implementaciones con el propósito de evaluar el desempeño de las técnicas basadas en redes neuronales convolucionales y los resultados fueron prometedores en cuanto a la tasa de aciertos, aunque aún no tenía un desempeño en tiempo real. Asimismo se decidió continuar con la utilización de estas técnicas imaginando que la reducción en los tiempos de procesamiento mejoraría en el corto plazo.

- *Perceptrón multicapa*: El estudio de redes neuronales convolucionales para la detección de manos y la estimación de 21 keypoints que forman su esqueleto, condujo a la utilización de las redes neuronales del tipo perceptrón multicapa logrando identificar los gestos de las manos a través de la información de dichos keypoints.

- *Creación de un dataset propio y entrenamiento de la red*: Las sucesivas implementaciones permitieron ajustar poco a poco la red perceptrón multicapa y posibilitó obtener información para la creación de un dataset robusto. Repetidamente se fueron agregando muestras al dataset, entrenando la red, ajustando sus hiperparámetros, tomando métricas, implementando el modelo y realizando los experimentos con el prototipo para avanzar hacia un mejor desempeño en cada iteración.

- *Evaluación del modelo*: Luego de tener implementado el modelo definitivo se realizaron las pruebas con datasets de terceros a fin de valorar el modelo frente a muestras ajenas y diversas.

- *Diseño de un controlador con gestos de las manos*: Luego de alcanzar una tasa de aciertos del 99,87 % con el modelo creado, se desarrolló el controlador llamado Hand Controller.

6.2. Contribuciones originales de esta tesis

Las contribuciones principales de esta tesis son las siguientes:

- *Arquitectura de red del tipo perceptrón multicapa para el reconocimiento de los gestos de las manos*: La arquitectura de red propuesta está inspirada en la arquitectura PointNet, la cual tiene mayor cantidad de capas y está diseñada para clasificar nube de puntos. En esta tesis, se propone disminuir la cantidad de capas y reducir considerablemente la dimensión de los datos de entrada, necesitando únicamente los 21 keypoints.

- *Modelo entrenado para reconocer 9 gestos*: La creación de un dataset propio permitió entrenar la red con 9 gestos y se realizaron más de 100 procesos de entrenamiento con distintos ajustes en los hiperparámetros. Finalmente se seleccionó el modelo con mayor tasa de aciertos, alcanzando el 99,87 %.

- *Capa de transformación y normalización*: Previo a ingresar los datos a la red, los 21 keypoints de las manos son sometidos a ciertas operaciones de normalización que logran generalizar el modelo para datos de entrada variados, lo cual permite mantener un buen desempeño cuando se utilizan gestos de distintas personas y en diversos escenarios.

- *Implementación del modelo*: Se utilizó el modelo entrenado para desarrollar un sistema de mando, llamado Hand Controller, que permite controlar una computadora con los gestos de las manos utilizando cámaras RGB de bajo coste.

6.3. Conclusiones

Durante el desarrollo de la propuesta de esta tesis se observó la importancia de la creación de prototipos que, en tiempo real, permitieran observar y valorar los resultados alcanzados en cada etapa. Es decir, siempre que sea posible, es beneficioso experimentar con la cámara RGB capturando las imágenes y procesando en tiempo real. Esto permite ajustar rápidamente los parámetros involucrados como también tener creatividad a la hora de vincular los conocimientos adquiridos en la exploración del estado de arte con la propuesta que se está desarrollando.

Disponer de estos prototipos para la visualización y evaluación del desempeño en tiempo real permitió identificar comportamientos que otorgaron capacidad para definir las operaciones de transformación y normalización de los datos, que son de las contribuciones más valiosas de esta tesis. Con estas operaciones no sólo se alcanza una elevada tasa de aciertos en la detección de gestos dentro de contextos[1] conocidos sino también en aquellos enteramente ajenos.

Otra de las conclusiones importantes es que una nube de puntos de baja densidad, tal como un conjunto de escasos puntos 3D del esqueleto de la mano, pueden ser suficientes para representar dicho esqueleto y la forma que adquiere el mismo. Es decir, una pequeña cantidad de puntos 3D ubicados en las articulaciones de un esqueleto son suficientes para representar su forma. Esta afirmación también fue validada para el esqueleto del cuerpo humano, ya que se realizaron unos primeros experimentos, planeando trabajos futuros, con una pequeña cantidad de puntos 3D ubicados en las articulaciones del cuerpo logrando una aceptable tasa de aciertos en las predicciones realizadas.

6.4. Trabajos futuros

Llegando al final del desarrollo de esta propuesta de tesis, se realizaron algunos experimentos simples con los keypoints extraídos del esqueleto del cuerpo humano y de la malla del rostro. Para ello se utilizó MediaPipe Pose que entrega 33 keypoints del cuerpo y MediaPipe Face Mesh que entrega 468 keypoints del rostro. Se creó un dataset de unas pocas muestras con 3 poses del cuerpo (brazos arriba de la cabeza, brazos relajados al costado del cuerpo y brazos cruzados) y se entrenó la misma red perceptrón multicapa con las correspondientes modificaciones para que la entrada a la red sea de (33, 3) en lugar de (21, 3). El modelo obtenido fue probado en tiempo real y los resultados fueron, en su mayoría, acertados. De la misma manera se creó un dataset con 3 expresiones faciales (expresión neutral, sonrisa y sorpresa), se entrenó la red con las modificaciones para una entrada de (468, 3) y los resultados también fueron aceptables.

En base a estos experimentos se propone, para trabajos futuros, utilizar esta misma arquitectura de red para el reconocimiento de las expresiones faciales y las posturas del cuerpo. De esta manera, es posible ampliar los mecanismos de detección de la intención de interacción que ya se encuentran implementados en Hand Controller. Para poner un

[1] Contexto hace referencia a las circunstancias que se producen en torno a la adquisición de las imágenes que se capturan para su procesamiento, como ser la iluminación, la cámara utilizada, la persona que realiza los gestos, la distancia hacia la cámara, entre otros.

ejemplo de aplicación se puede mencionar que una persona que desea interactuar con un sistema a través de los gestos de las manos, en general, no tiene los brazos relajados al costado del cuerpo sino que tendría uno de los brazos levantados. Esta intención de la persona por interactuar con el sistema podría ser detectado identificando la postura de los brazos. De este modo, se podría utilizar el reconocimiento de gestos de las manos únicamente cuando un brazo se encuentre levantado, suponiendo con ello que la persona tiene intenciones de interactuar con el sistema.

Bibliografía

Bibliografía

[Afifi, 2019] Afifi, M. (2019). 11k hands: gender recognition and biometric identification using a large dataset of hand images. *Multimedia Tools and Applications*.

[Bai et al., 2018] Bai, S., Kolter, J. Z., and Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling.

[Bambach et al., 2015] Bambach, S., Lee, S., Crandall, D. J., and Yu, C. (2015). Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *The IEEE International Conference on Computer Vision (ICCV)*.

[Bao et al., 2017] Bao, P., Maqueda, A. I., del Blanco, C. R., and García, N. (2017). Image database for tiny hand gesture recognition. `https://sites.google.com/view/handgesturedb/home`. [Online; accessed 2-July-2021].

[Bayer and Faigl, 2019] Bayer, J. and Faigl, J. (2019). On autonomous spatial exploration with small hexapod walking robot using tracking camera intel realsense t265. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–6.

[Cantareira and Paulovich, 2020] Cantareira, G. D. and Paulovich, F. V. (2020). A Generic Model for Projection Alignment Applied to Neural Network Visualization. In Turkay, C. and Vrotsou, K., editors, *EuroVis Workshop on Visual Analytics (EuroVA)*. The Eurographics Association.

[Chang et al., 2015] Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015). Shapenet: An information-rich 3d model repository.

[Chang et al., 2018] Chang, J. Y., Moon, G., and Lee, K. M. (2018). V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5079–5088.

[Chao et al., 2021] Chao, Y.-W., Yang, W., Xiang, Y., Molchanov, P., Handa, A., Tremblay, J., Narang, Y. S., Van Wyk, K., Iqbal, U., Birchfield, S., Kautz, J., and Fox, D. (2021). DexYCB: A benchmark for capturing hand grasping of objects. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Chen et al., 2019] Chen, Y., Tu, Z., Ge, L., Zhang, D., Chen, R., and Yuan, J. (2019). So-handnet: Self-organizing network for 3d hand pose estimation with semi-supervised learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6960–6969.

[Chen et al., 2021] Chen, Y., Tu, Z., Kang, D., Bao, L., Zhang, Y., Zhe, X., Chen, R., and Yuan, J. (2021). Model-based 3d hand reconstruction via self-supervised learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10446–10455.

[Chhibber et al., 2021] Chhibber, N., Surale, H. B., Matulic, F., and Vogel, D. (2021). Typealike: Near-keyboard hand postures for expanded laptop interaction. *Proceedings of the ACM on Human-Computer Interaction*, 5(ISS):1–20.

[Corona et al., 2022] Corona, E., Hodan, T., Vo, M., Moreno-Noguer, F., Sweeney, C., Newcombe, R., and Ma, L. (2022). Lisa: Learning implicit shape and appearance of hands.

[Corona et al., 2020] Corona, E., Pumarola, A., Alenya, G., Moreno-Noguer, F., and Rogez, G. (2020). Ganhand: Predicting human grasp affordances in multi-object scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5031–5041.

[Damen et al., 2022] Damen, D., Doughty, H., Farinella, G. M., Furnari, A., Ma, J., Kazakos, E., Moltisanti, D., Munro, J., Perrett, T., Price, W., and Wray, M. (2022). Rescaling egocentric vision: Collection, pipeline and challenges for epic-kitchens-100. *International Journal of Computer Vision (IJCV)*, 130:33–55.

[Damindarov et al., 2021] Damindarov, R., Fam, C. A., Boby, R. A., Fahim, M., Klimchik, A., and Matsumaru, T. (2021). A depth camera-based system to enable touchless interaction using hand gestures. In *2021 International Conference "Nonlinearity, Information and Robotics"(NIR)*, pages 1–7.

[de Amorim and Hennig, 2015] de Amorim, R. C. and Hennig, C. (2015). Recovering the number of clusters in data sets with noise features using feature rescaling factors. *Information Sciences*, 324:126–145.

[De Smedt et al., 2016] De Smedt, Q., Wannous, H., and Vandeborre, J.-P. (2016). Skeleton-based dynamic hand gesture recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1206–1214.

[De Smedt et al., 2017] De Smedt, Q., Wannous, H., Vandeborre, J.-P., Guerry, J., Saux, B. L., and Filliat, D. (2017). 3d hand gesture recognition using a depth and skeletal dataset: Shrec'17 track. In *Proceedings of the Workshop on 3D Object Retrieval*, 3Dor '17, page 33–38, Goslar, DEU. Eurographics Association.

[Devlin et al., 2019] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. abs/1810.04805.

[Du et al., 2019] Du, K., Lin, X., Sun, Y., and Ma, X. (2019). Crossinfonet: Multi-task information sharing based hand pose estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9888–9897.

[Fadzli and Ismail, 2019] Fadzli, F. E. and Ismail, A. W. (2019). Voxar: 3d modelling editor using real hands gesture for augmented reality. In *2019 IEEE 7th Conference on Systems, Process and Control (ICSPC)*, pages 242–247.

[Garcia-Hernando et al., 2018] Garcia-Hernando, G., Yuan, S., Baek, S., and Kim, T.-K. (2018). First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*.

[Ge et al., 2019] Ge, L., Ren, Z., Li, Y., Xue, Z., Wang, Y., Cai, J., and Yuan, J. (2019). 3d hand shape and pose estimation from a single rgb image. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10825–10834.

[Ge et al., 2018] Ge, L., Ren, Z., and Yuan, J. (2018). Point-to-point regression pointnet for 3d hand pose estimation. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision – ECCV 2018*, pages 489–505, Cham. Springer International Publishing.

[Hasson et al., 2019] Hasson, Y., Varol, G., Tzionas, D., Kalevatykh, I., Black, M. J., Laptev, I., and Schmid, C. (2019). Learning joint reconstruction of hands and manipulated objects. In *CVPR*.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[Hsu et al., 2021] Hsu, C. P., Li, B., Solano-Rivas, B., Gohil, A. R., Chan, P. H., Moore, A. D., and Donzella, V. (2021). A review and perspective on optical phased array for automotive lidar. *IEEE Journal of Selected Topics in Quantum Electronics*, 27(1):1–16.

[Hu et al., 2020] Hu, F., He, P., Xu, S., Li, Y., and Zhang, C. (2020). Fingertrak: Continuous 3d hand pose tracking by deep learning hand silhouettes captured by miniature thermal cameras on wrist. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4:1–24.

[Hu et al., 2021a] Hu, H., Wang, W., Zhou, W., Zhao, W., and Li, H. (2021a). Model-aware gesture-to-gesture translation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16423–16432.

[Hu et al., 2021b] Hu, H., Zhao, W., Zhou, W., Wang, Y., and Li, H. (2021b). Signbert: Pre-training of hand-model-aware representation for sign language recognition. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11067–11076.

[Hu et al., 2021c] Hu, H., Zhou, W., Pu, J., and Li, H. (2021c). Global-local enhancement network for nmf-aware sign language recognition. 17(3).

[Iqbal et al., 2018] Iqbal, U., Molchanov, P., Breuel, T., Gall, J., and Kautz, J. (2018). Hand pose estimation via latent 2.5d heatmap regression. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision – ECCV 2018*, pages 125–143, Cham. Springer International Publishing.

[Khan and Borji, 2018] Khan, A. U. and Borji, A. (2018). Analysis of hand segmentation in the wild. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4710–4719.

[Kim et al., 2021] Kim, D. U., In Kim, K., and Baek, S. (2021). End-to-end detection and pose estimation of two interacting hands. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11169–11178.

[Kim et al., 2018] Kim, Y., An, S.-G., Lee, J., and Bae, S.-H. (2018). Agile 3d sketching with air scaffolding. In *CHI '18 Conference on Human Factors in Computing Systems*, pages 1–12.

[Kingma and Ba, 2015] Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations*.

[Kulon et al., 2020] Kulon, D., Guler, R. A., Kokkinos, I., Bronstein, M. M., and Zafeiriou, S. (2020). Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *CVPR Workshop on Computer Vision for Augmented and Virtual Reality*.

[Lai et al., 2015] Lai, C.-L., Huang, Y.-L., Liao, T.-K., Tseng, C.-M., Chen, Y.-F., and Erdenetsogt, D. (2015). A microsoft kinect-based virtual rehabilitation system to train balance ability for stroke patients. In *2015 International Conference on Cyberworlds (CW)*, pages 54–60.

[Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[Lee et al., 2018] Lee, J. H., An, S., Kim, Y., and Bae, S. (2018). Projective windows: Bringing windows in space to the fingertip. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*, pages 1–8. ACM.

[Li et al., 2020] Li, D., Rodriguez, C., Yu, X., and Li, H. (2020). Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 1459–1469.

[Li et al., 2010] Li, W., Zhang, Z., and Liu, Z. (2010). Action recognition based on a bag of 3d points. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 9–14.

[Li et al., 2021] Li, Y., Liu, M., and Rehg, J. (2021). In the eye of the beholder: Gaze and actions in first person video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

[Liao and Wang, 2019] Liao, J. and Wang, H. (2019). Gestures as intrinsic creativity support: Understanding the usage and function of hand gestures in computer-mediated group brainstorming. *Proc. ACM Hum. Comput. Interact.*, 3(GROUP):243:1–243:16.

[Lin et al., 2021] Lin, C., Li, C., Liu, Y., Chen, N., Choi, Y.-K., and Wang, W. (2021). Point2skeleton: Learning skeletal representations from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4277–4286.

[Liu et al., 2022a] Liu, D., Zhang, L., and Wu, Y. (2022a). Ld-congr: A large rgb-d video dataset for long-distance continuous gesture recognition. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3294–3302.

[Liu et al., 2020] Liu, J., Ding, H., Shahroudy, A., Duan, L.-Y., Jiang, X., Wang, G., and Kot, A. C. (2020). Feature boosting network for 3d pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):494–501.

[Liu et al., 2022b] Liu, S., Tripathi, S., Majumdar, S., and Wang, X. (2022b). Joint hand motion and interaction hotspots prediction from egocentric videos. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3272–3282.

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *ECCV (1)*, pages 21–37. Springer.

[Ma and Yang, 2020] Ma, W. and Yang, K. (2020). A fast reconstruction method of 3d object point cloud based on realsense d435. In *2020 39th Chinese Control Conference (CCC)*, pages 6650–6656.

[Malik et al., 2020] Malik, J., Abdelaziz, I., Elhayek, A., Shimada, S., Ali, S. A., Golyanik, V., Theobalt, C., and Stricker, D. (2020). Handvoxnet: Deep voxel-based network for 3d hand shape and pose estimation from a single depth map. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7111–7120.

[Matulic et al., 2020] Matulic, F., Arakawa, R., Vogel, B., and Vogel, D. (2020). Pensight: Enhanced interaction with a pen-top camera. In *CHI '20 Conference on Human Factors in Computing Systems*, pages 1–14.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. In *The Bulletin of Mathematical Biophysics*, volume 5, page 115–133.

[Memo et al., 2015] Memo, A., Minto, L., and Zanuttigh, P. (2015). Exploiting Silhouette Descriptors and Synthetic Data for Hand Gesture Recognition. `https://lttm.dei.unipd.it/downloads/gesture/`. [Online; accessed 2-July-2021].

[Menze and Geiger, 2015] Menze, M. and Geiger, A. (2015). Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Min et al., 2019] Min, Y., Chai, X., Zhao, L., and Chen, X. (2019). Flickernet: Adaptive 3d gesture recognition from sparse point clouds. In *BMVC*, page 105.

[Min et al., 2020] Min, Y., Zhang, Y., Chai, X., and Chen, X. (2020). An efficient pointlstm for point clouds based gesture recognition. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5760–5769.

[Mittal et al., 2019] Mittal, A., Kumar, P., Roy, P. P., Balasubramanian, R., and Chaudhuri, B. B. (2019). A modified lstm model for continuous sign language recognition using leap motion. *IEEE Sensors Journal*, 19(16):7056–7063.

[Molchanov et al., 2016] Molchanov, P., Yang, X., Gupta, S., Kim, K., Tyree, S., and Kautz, J. (2016). Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4207–4215.

[Moon et al., 2020] Moon, G., Yu, S.-I., Wen, H., Shiratori, T., and Lee, K. M. (2020). Interhand2.6m: A dataset and baseline for 3d interacting hand pose estimation from a single rgb image. In *European Conference on Computer Vision (ECCV)*.

[Mostafa et al., 2021] Mostafa, S., Mondal, D., Beck, M., Bidinosti, C., Henry, C., and Stavness, I. (2021). Visualizing feature maps for model selection in convolutional neural networks. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1362–1371.

[Mueller et al., 2018] Mueller, F., Bernard, F., Sotnychenko, O., Mehta, D., Sridhar, S., Casas, D., and Theobalt, C. (2018). Ganerated hands for real-time 3d hand tracking from monocular rgb. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 49–59.

[Muhammad and Kim, 2020] Muhammad, S. and Kim, G. (2020). Visual object detection based lidar point cloud classification. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 438–440.

[Nam et al., 2021] Nam, G., Heo, M., Oh, S. W., Lee, J.-Y., and Kim, S. J. (2021). Polygonal point set tracking. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5565–5574.

[Nguyen et al., 2019] Nguyen, X. S., Brun, L., Lézoray, O., and Bougleux, S. (2019). A neural network based on spd manifold learning for skeleton-based hand gesture recognition. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12028–12037.

[Ono et al., 2020] Ono, T., Akasaka, S., Ohara, H., and Kanamaru, T. (2020). Developing education support system using interactive floor interface. In *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, pages 141–144.

[Palieri et al., 2020] Palieri, M., Morrell, B., Thakur, A., Ebadi, K., Nash, J., Chatterjee, A., Kanellakis, C., Carlone, L., Guaragnella, C., and Agha-mohammadi, A. (2020). Locus: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time. *IEEE Robotics and Automation Letters*, pages 1–1.

[Patil et al., 2022] Patil, S., Kodarlikar, S., Marathe, S., and Chandran, D. V. (2022). Touchless ecosystem using hand gestures. *International Research Journal of Engineering and Technology*, Volume 9, Issue 4.

[Pei et al., 2022] Pei, S., Chen, A., Lee, J., and Zhang, Y. (2022). Hand interfaces: Using hands to imitate objects in ar/vr for expressive interactions. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, New York, NY, USA. Association for Computing Machinery.

[Perazzi et al., 2016] Perazzi, F., Pont-Tuset, J., McWilliams, B., Van Gool, L., Gross, M., and Sorkine-Hornung, A. (2016). A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*.

[Qi et al., 2017] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660.

[Ren et al., 2022] Ren, P., Sun, H., Hao, J., Wang, J., Qi, Q., and Liao, J. (2022). Mining multi-view information: A strong self-supervised framework for depth-based 3d hand pose and mesh estimation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20523–20533.

[Romero et al., 2017] Romero, J., Tzionas, D., and Black, M. J. (2017). Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6).

[Rosenblatt, 1957] Rosenblatt, F. (1957). A perceiving and recognizing automaton. In *Cornell Aeronautical Laboratory*.

[Rousseeuw, 1987] Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

[Seo and Joo, 2020] Seo, H. and Joo, S. (2020). Influence of preprocessing and augmentation on 3d point cloud classification based on a deep neural network: Pointnet. In *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, pages 895–899.

[Taheri et al., 2020] Taheri, O., Ghorbani, N., Black, M. J., and Tzionas, D. (2020). GRAB: A dataset of whole-body human grasping of objects. In *European Conference on Computer Vision (ECCV)*.

[Thakur, 2019] Thakur, A. (2019). American Sign Language Dataset for Image Classifcation. `https://www.kaggle.com/ayuraj/asl-dataset`. [Online; accessed 2-July-2021].

[Ultraleap, 2022] Ultraleap (2022). Product Catalogue: Interactive Kiosks Range. `https://www.ultraleap.com/enterprise/touchless-experiences/touchfree-solution`. [Online; accessed 4-Ago-2022].

[Vaezi Joze and Koller, 2019] Vaezi Joze, H. and Koller, O. (2019). Ms-asl: A large-scale data set and benchmark for understanding american sign language. In *The British Machine Vision Conference (BMVC)*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[Velayudhan and Gireeshkumar, 2015] Velayudhan, A. and Gireeshkumar, T. (2015). *An Autonomous Obstacle Avoiding and Target Recognition Robotic System Using Kinect*, volume 308, pages 643–649. a.

[Wang et al., 2021] Wang, Z. J., Turko, R., Shaikh, O., Park, H., Das, N., Hohman, F., Kahng, M., and Polo Chau, D. H. (2021). Cnn explainer: Learning convolutional neural networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1396–1406.

[Weng et al., 2021] Weng, Y., Yu, C., Shi, Y., Zhao, Y., Yan, Y., and Shi, Y. (2021). Facesight: Enabling hand-to-face gesture interaction on ar glasses with a downward-facing camera vision. In *CHI '21 Conference on Human Factors in Computing Systems*, pages 1–14.

[Wu et al., 2015] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shape modeling. In *Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015)*.

[Yang et al., 2020] Yang, S., Liu, J., Lu, S., Er, M. H., and Kot, A. C. (2020). Collaborative learning of gesture recognition and 3d hand pose estimation with multi-order feature analysis. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III*, page 769–786, Berlin, Heidelberg. Springer-Verlag.

[Yang et al., 2019] Yang, Z., Liu, S., Hu, H., Wang, L., and Lin, S. (2019). Reppoints: Point set representation for object detection. In *The IEEE International Conference on Computer Vision (ICCV)*.

[Ye and Kim, 2018] Ye, Q. and Kim, T.-K. (2018). Occlusion-aware hand pose estimation using hierarchical mixture density network. In Ferrari, V., Hebert, M., Sminchisescu, C., and Weiss, Y., editors, *Computer Vision – ECCV 2018*, pages 817–834, Cham. Springer International Publishing.

[Yuan et al., 2018] Yuan, S., Garcia-Hernando, G., Stenger, B., Moon, G., Chang, J. Y., Lee, K. M., Molchanov, P., Kautz, J., Honari, S., Ge, L., Yuan, J., Chen, X., Wang, G., Yang, F., Akiyama, K., Wu, Y., Wan, Q., Madadi, M., Escalera, S., Li, S., Lee, D., Oikonomidis, I., Argyros, A., and Kim, T.-K. (2018). Depth-based 3d hand pose estimation: From current achievements to future goals. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2636–2645.

[Zhang et al., 2020] Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L., and Grundmann, M. (2020). Mediapipe hands: On-device real-time hand tracking.

[Zhang and Yang, 2019] Zhang, W. and Yang, D. (2019). Lidar-based fast 3d stockpile modeling. In *2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, pages 703–707.

[Zhao et al., 2022] Zhao, W., Liu, X., Zhong, Z., Jiang, J., Gao, W., Li, G., and Ji, X. (2022). Self-supervised arbitrary-scale point clouds upsampling via implicit neural representation. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1989–1997.

[Zhirong Wu et al., 2015] Zhirong Wu, Song, S., Khosla, A., Fisher Yu, Linguang Zhang, Xiaoou Tang, and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920.

[Zhou et al., 2020] Zhou, Q., Sykes, S., Fels, S., and Kin, K. (2020). Gripmarks: Using hand grips to transform in-hand objects into mixed reality input. In *CHI '20: Conference on Human Factors in Computing Systems*, pages 1–11.

[Zimmermann et al., 2019] Zimmermann, C., Ceylan, D., Yang, J., Russel, B., Argus, M., and Brox, T. (2019). Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *IEEE International Conference on Computer Vision (ICCV)*.

Este documento ha sido generado usando LaTeX.

Todas las figuras son de elaboración propia, salvo en las que se indica la fuente.

La paleta de colores para las Figuras está inspirado en las pinturas al óleo de Joan Miró.

Reconocimiento de posturas de las manos
para la Interacción Natural Humano-Computadora
en Ambientes Inteligentes a través de cámara RGB

César Alejandro Osimani

Centro de Investigación Aplicada y Desarrollo
en Informática y Telecomunicaciones (CIADE-IT)

Universidad Blas Pascal

Córdoba, noviembre de 2023