

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA



TESIS DE GRADO

**"RECONOCIMIENTO E INTERPRETACIÓN DEL ALFABETO
DACTILOLÓGICO DE LA LENGUA DE SEÑAS MEDIANTE
TECNOLOGÍA MÓVIL Y REDES NEURONALES ARTIFICIALES"**

PARA OPTAR EL TÍTULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN: INGENIERÍA DE SISTEMAS INFORMÁTICOS

POSTULANTE: REYNALDO ANTONIO AQUINO CASTRO
TUTOR METODOLÓGICO: M.Sc. GROVER ALEX RODRÍGUEZ RAMÍREZ
ASESOR: M.Sc. GERMÁN HUANCA TICONA

LA PAZ – BOLIVIA

2018



UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA



LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

- a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
- b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
- c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

TODOS LOS DERECHOS RESERVADOS. EL USO NO AUTORIZADO DE LOS CONTENIDOS PUBLICADOS EN ESTE SITIO DERIVARA EN EL INICIO DE ACCIONES LEGALES CONTEMPLADOS EN LA LEY DE DERECHOS DE AUTOR.

DEDICATORIA

A mi querida madre Julia Castro por todo el trabajo y sacrificio que realizó estos años para darme todo lo necesario y así poder llegar y concretar esta importante etapa de mi vida.

AGRADECIMIENTOS

A mi madre, que con su demostración de una madre ejemplar me ha enseñado a no desfallecer ni rendirme ante nada y siempre perseverar ante las más grandes dificultades y obstáculos que se nos presentan en la vida, muchas gracias por esos sabios consejos.

A mis hermanas Yaquelin y Carolina por toda la colaboración y por ser las personas más importantes en mi vida ya que juntos hemos pasado momentos inolvidables.

A mi abuelita Paola y a mis tíos Emilio, Cesar, Cilda y Sandra por su apoyo incondicional y por demostrarme la gran fe que tienen en mí.

A mi tutor M. Sc. Grover Alex Rodríguez Ramírez, por su valiosa guía y asesoramiento en la realización de la presente tesis de grado.

A mi asesor M. Sc. Germán Huanca Ticona, por todos los consejos, correcciones, revisiones y tiempo que me brindó para el desarrollo de la presente tesis de grado.

A si mismo gracias a Joselin por su gran apoyo y a todos mis amigos que ayudaron directa e indirectamente en la realización de este trabajo de grado

RESUMEN

Los procesos comunicativos son de vital importancia para el desarrollo de cualquier individuo y de una sociedad, estos permiten manifestar ideas, sentimientos, necesidades entre otras emociones. La lengua de señas ayuda a las personas sordas a llevar una vida con comunicación, ya que es el medio con el que podemos comunicarnos personas oyentes y no oyentes. Este lenguaje que podría incluir en mayor medida a las personas con sordera o invalidez del habla, es conocido por un número muy reducido de individuos sin estas discapacidades de nuestra sociedad, lo que hace difícil la comunicación con el resto de la población que no conocen de la lengua de señas. Esto puede llegar a provocar sentimientos de soledad, frustración, depresión y baja autoestima en la persona que padece sordera.

La lengua de señas está compuesta por un conjunto estructurado de gestos, movimientos, posturas y expresiones faciales que corresponden a letras o palabras. Para poder representar las diferentes letras del alfabeto se hace uso del alfabeto dactilológico, el cual sirve para comunicarse cuando no se conoce o no existe una seña.

El presente trabajo de grado presenta un sistema de reconocimiento e interpretación del alfabeto dactilológico mediante una red neuronal artificial desarrollada en TensorFlow y siguiendo la metodología de visión artificial apoyada por la librería OpenCV. Para la construcción de la red neuronal se utilizó la técnica de transferencia de aprendizaje reentrenando una red neuronal convolucional previamente entrenada. Posteriormente se optimizó este modelo de red neuronal para ser implementado en una aplicación para dispositivos móviles con sistema operativo Android.

Finalmente se hicieron experimentos con la aplicación móvil para evaluar la fiabilidad del sistema, del cual, se obtuvieron resultados satisfactorios bajo ciertas condiciones controladas de distancia e iluminación.

ABSTRACT

The communicative processes are of vital importance for the development of any individual and of a society, these allow to express ideas, feelings, and needs among other emotions. Sign language helps deaf people lead a life with communication, as it is the means by which we can communicate hearing and non-hearing people. This language, which could include more people with deafness or speech disability, is known by a very small number of individuals without these disabilities in our society, which makes it difficult to communicate with the rest of the population that they do not know about. The sign language. This can lead to feelings of loneliness, frustration, depression and low self-esteem in the person suffering from deafness.

Sign language is composed of a structured set of gestures, movements, postures and facial expressions that correspond to letters or words. In order to represent the different letters of the alphabet, the fingerprint alphabet is used, which is used to communicate when there is no known or no sign. The present work of degree presents a system of recognition and interpretation of the dactylographic alphabet through an artificial neural network developed in TensorFlow and following the methodology of artificial vision supported by the OpenCV library. For the construction of the neural network, the learning transfer technique was used, retraining a previously trained convolutional neuronal network. Subsequently this model of neural network was optimized to be implemented in an application for mobile devices with Android operating system.

Finally experiments were made with the mobile application to evaluate the reliability of the system, of which satisfactory results were obtained under certain controlled conditions of distance and illumination.

ÍNDICE

CAPÍTULO I.....	1
MARCO INTRODUCTORIO	1
1.1 INTRODUCCIÓN	1
1.2 ANTECEDENTES.....	2
1.3 PLANTEAMIENTO DEL PROBLEMA.....	8
1.3.1 PROBLEMA CENTRAL.....	8
1.3.2 PROBLEMAS SECUNDARIOS	8
1.4 DEFINICIÓN DE OBJETIVOS	9
1.4.1 OBJETIVO GENERAL	9
1.4.2 OBJETIVOS ESPECÍFICOS	9
1.5 HIPÓTESIS.....	10
1.6 JUSTIFICACIÓN.....	10
1.6.1 JUSTIFICACIÓN ECONÓMICA.....	10
1.6.2 JUSTIFICACIÓN SOCIAL	10
1.6.3 JUSTIFICACIÓN CIENTIFICA.....	10
1.7 ALCANCES Y LÍMITES	11
1.7.1 ALCANCES	11
1.7.2 LÍMITES	11
1.8 APORTE.....	12
1.8.1 PRÁCTICO	12
1.8.2 TEÓRICO	12
CAPÍTULO II	13
MARCO TEÓRICO	13
2.1 INTELIGENCIA ARTIFICIAL	13
2.1.1 EL ASCENSO DE MACHINE LEARNING.....	14
2.2 REDES NEURONALES ARTIFICIALES.....	15
2.2.1 FUNDAMENTOS BIOLÓGICOS DE LAS REDES NEURONALES NATURALES	
15	
2.2.2 ELEMENTOS FUNCIONALES DE LA NEURONA ARTIFICIAL	17
2.2.3 FUNCIONES DE ACTIVACIÓN.....	19
2.2.3.1 FUNCIÓN ESCALÓN	19

2.2.3.2	FUNCIÓN LINEAL Y MIXTA	20
2.2.3.3	FUNCIÓN TANGENTE HIPERBÓLICA	20
2.2.3.4	FUNCIÓN SIGMOIDAL	21
2.2.3.5	FUNCIÓN DE GAUSS	22
2.2.4	APRENDIZAJE.....	22
2.2.5	APRENDIZAJE SUPERVISADO	22
2.2.5.1	PERCEPTRÓN	23
2.2.5.2	ADALINE.....	26
2.2.5.3	MADALINE	27
2.2.6	APRENDIZAJE NO SUPERVISADO	28
2.3	REDES NEURONALES CONVOLUCIONALES	28
2.3.1	ESTRUCTURA DE REDES NEURONALES CONVOLUCIONALES	29
2.4	VISIÓN ARTIFICIAL	35
2.4.1	ETAPAS UN PROCESO DE VISIÓN ARTIFICIAL	36
2.4.2	ADQUISICIÓN DE IMÁGENES	38
2.4.2.1	MODELOS DE CAPTURA DE IMÁGENES	39
2.4.2.2	LA DIGITALIZACIÓN.....	39
2.4.2.3	REPRESENTACIÓN DE LA IMAGEN Y ESTRUCTURAS DE DATOS	39
2.4.3	PREPROCESADO	41
2.4.3.1	FILTRADO ESPACIAL.....	42
2.4.4	SEGMENTACIÓN.....	46
2.4.5	PARAMETRIZACIÓN	47
2.4.6	RECONOCIMIENTO E INTERPRETACIÓN.....	48
2.5	OPEN COMPUTER VISION LIBRARY.....	48
2.5.1	ESTRUCTURA Y CARACTERÍSTICAS DE LA LIBRERÍA OPENCV	49
2.5.2	HERRAMIENTAS DE LA LIBRERÍA OPENCV	49
2.5.3	OPENCV MANAGER	50
2.6	TENSORFLOW	50
2.6.1	FLUJO DE TRABAJO	51
2.6.2	TENSOR.....	51
2.6.3	GRAPH.....	52
2.6.4	TENSORBOARD.....	53

2.6.5	SESSION	53
2.6.6	ENTRENAMIENTO	54
2.7	SISTEMA OPERATIVO ANDROID.....	55
2.7.1	CARACTERÍSTICAS.....	55
2.7.2	ARQUITECTURA	56
2.7.2.1	KERNEL DE LINUX	56
2.7.2.2	CAPA DE ABSTRACCIÓN DE HARDWARE (HAL)	56
2.7.2.3	TIEMPO DE EJECUCIÓN DE ANDROID.....	57
2.7.2.4	BIBLIOTECAS C/C++ NATIVAS	58
2.7.2.5	FRAMEWORK JAVA API.....	59
2.7.2.6	APPS DEL SISTEMA	59
2.7.3	API DE REDES NEURONALES	60
2.8	LENGUA DE SEÑAS	61
2.8.1	ALFABETO DACTIOLÓGICO.....	61
2.8.2	DACTILOGRÍA	63
CAPÍTULO III	64
	DISEÑO METODOLÓGICO	64
3.1	INTRODUCCIÓN	64
3.2	MATERIAL REQUERIDO	66
3.2.1	EQUIPO UTILIZADO	66
3.2.2	SOFTWARE REQUERIDO.....	66
3.3	ADQUISICIÓN DE LA IMAGEN	67
3.3.1	DISPOSITIVO DE CAPTURA.....	67
3.3.2	DESARROLLO DE LA INTERFAZ DEL SISTEMA.....	69
3.3.2.1	USO DE PERMISOS.....	69
3.3.2.2	INTERACCIÓN CON LA CAMARA DEL DISPOSITIVO.....	70
3.4	PREPROCESADO.....	71
3.4.1	EXTRACCIÓN DE ÁREAS DE INTERÉS	72
3.4.2	ELIMINACIÓN DE RUIDO.....	72
3.5	SEGMENTACIÓN	73
3.6	PARAMETRIZACIÓN	73
3.6.1	EXTRACCIÓN DE CARACTERÍSTICAS IMPORTANTES O RELEVANTES DE LA IMAGEN.....	73

3.7	RECONOCIMIENTO	74
3.7.1	MODELO DE RED NEURONAL CONVOLUCIONAL	74
3.7.1.1	COMPONENTES PRINCIPALES DE LA RED NEURONAL CONVOLUCIONAL	76
3.8	INTERPRETACIÓN.....	80
3.9.	APLICACIÓN MÓVIL.....	81
3.9.1.	OPTIMIZAR EL MODELO	82
3.9.1.1.	LIBRERÍAS LIMITADAS EN DISPOSITIVOS MÓVILES	82
3.9.1.2.	OPTIMIZAR PARA LA INFERENCIA	82
3.9.1.3.	VERIFICAR EL MODELO OPTIMIZADO.....	83
3.9.2.	CONFIGURACIÓN DE LA APLICACIÓN ANDROID	83
3.10.	DESCRIPCION DE LA APLICACIÓN MÓVIL	86
3.11.	PRUEBAS	87
CAPÍTULO IV	89	
ANÁLISIS DE RESULTADOS Y DEMOSTRACIÓN DE HIPOTESIS		89
4.	INTRODUCCIÓN	89
4.1	ANÁLISIS DE DATOS	89
4.2	DEMOSTRACIÓN DE LA HIPÓTESIS	90
4.2.1	CONTRASTE DE RACHAS DE WALD-WOLFOWITZ	91
4.2.2	DESARROLLO DE LA DEMOSTRACIÓN DE LA HIPÓTESIS.....	92
CAPÍTULO V	97	
CONCLUSIONES Y RECOMENDACIONES.....		97
4.	CONCLUSIONES	97
5.	RECOMENDACIONES.....	98
BIBLIOGRAFÍA	99	

ÍNDICE DE FIGURAS

Figura 2.1: Elementos neurales	16
Figura 2.2: Esquema básico de una RNA.....	17
Figura 2.3: Esquema de una neurona artificial.	17
Figura 2.4: Función de activación escalón.....	19
Figura 2.5: Función de activación lineal y mixta.	20
Figura 2.6: Función tangente hiperbólica	21
Figura 2.7: Función sigmoidal.	21
Figura 2.8: Función de activación gaussiana	22
Figura 2.9: Perceptrón clásico propuesto por McCulloch – Pitts.....	23
Figura 2.10: Clasificación de patrones con el perceptrón.....	25
Figura 2.11: Sistema de entrenamiento para una red ADALINE.	26
Figura 2.12: ADALINE (con salida de activación lineal).	26
Figura 2.13: Una red neuronal convolucional 2D.....	29
Figura 2.14: Ejemplo de una red neuronal convolucional 2D paso 1	29
Figura 2.15: Ejemplo de una red neuronal convolucional 2D paso 2	30
Figura 2.16: Ejemplo de una red neuronal convolucional 2D paso 3	30
Figura 2.17: Ejemplo de una red neuronal convolucional 2D paso 4	31
Figura 2.18: Ejemplo de una red neuronal convolucional 2D paso 5	31
Figura 2.19: Ejemplo de una red neuronal convolucional 2D paso 6	32
Figura 2.20: Ejemplo de una red neuronal convolucional 2D paso 7	33
Figura 2.21: Ejemplo de una red neuronal convolucional 2D paso 8	33
Figura 2.22: Ejemplo de una red neuronal convolucional 2D paso 9	34
Figura 2.23: Diagrama de bloques de las etapas de un sistema de visión artificial	37
Figura 2.24: Esquema de un fichero gráfico.....	40
Figura 2.25: Esquema de funcionamiento de un filtro.	42
Figura 2.26: Aplicación de un filtrado espacial paso bajo.....	44
Figura 2.27: Filtrado paso alto de Sobel en la dirección x en valor absoluto. El resultado se presenta con el valor de los píxeles invertidos.....	46
Figura 2.28: Ejemplo TensorBoard.....	53
Figura 2.29: Arquitectura de Software de Android.....	57
Figura 2.30 Alfabeto Dactilológico de la Lengua de Señas Boliviana.....	62

Figura 2.31: Representación manual de la letra “A”	63
Figura 3.1: Etapas del sistema de Visión Artificial para el Reconocimiento e Interpretación del alfabeto dactilológico.....	65
Figura 3.2: Interfaz de la aplicación móvil	71
Figura 3.3: Arquitectura Inception v3.	75
Figura 3.4: Capas convolucionales.....	76
Figura 3.5: Pooling.....	77
Figura 3.6: Capas totalmente conectada	77
Figura 3.7: Reentrenamiento de la red neuronal.	80
Figura 3.8: Primera sección de la aplicación.	81
Figura 3.9: Estructura de archivos para Tensorflow en Android.	84
Figura 3.10: Estructura de archivos para el modelo previamente entrenado.	85
Figura 3.11: Requerimiento de permisos de la aplicación.....	86
Figura 3.12: Ejecución de la aplicación móvil.....	87
Figura 3.13: Herramienta Tensorboard	88
Figura 4.1: Exactitud de la red neuronal.....	89
Figura 4.2: Exactitud final de la red neuronal.....	95

ÍNDICE DE TABLAS

Tabla 2.1: Diferentes formatos para ficheros gráficos y características principales.	41
Tabla 2.2: Comparación de reconocimiento e interpretación por una persona y la aplicación móvil	92

CAPÍTULO I

MARCO INTRODUCTORIO

1.1 INTRODUCCIÓN

La comunicación es un proceso de intercambio de información a través de un sistema compartido de signos y normas semánticas, siendo este un pilar fundamental para la existencia de una sociedad, ya que facilita la transmisión de conocimiento, ideas, sentimientos, pensamientos, costumbres y valores culturales. La comunicación entre personas se la realiza predominantemente por el lenguaje oral, que es una forma natural y espontánea de comunicación. Sin embargo, por distintas razones no todas las personas pueden comunicarse mediante el lenguaje oral.

Las personas sordas son aquellas que tienen una deficiencia total o profunda de la audición, es por este motivo que tienen una dificultad para poder comunicarse con las demás personas por medio del lenguaje oral. Como menciona Raynond Carhart (Hearing and Deafness, 1947). “El habla, normalmente, se controla por el oído”. Quien pierde el oído, fácilmente pierde el habla. Es por esta razón que las personas sordas utilizan la lengua de señas como principal método de comunicación.

La lengua de señas está compuesta por un conjunto estructurado de gestos, movimientos, posturas y expresiones faciales que corresponden a letras o palabras. Para poder representar las diferentes letras del alfabeto se hace uso del alfabeto dactilológico que, aunque en sí mismo no son señas sino una representación manual para las diferentes letras, el cual sirve para comunicarse cuando no se conoce o no existe una señal.

La lengua de señas no tiene una estructura general a nivel mundial, por lo que cada país tiene su propia lengua de señas, en el caso de nuestro país es denominado la Lengua de Señas Boliviana (LSB)

Debido a que muchas personas no tienen conocimiento de la lengua de señas, las personas sordas tienen dificultades al comunicarse con estas personas, especialmente cuando se encuentran en medio de muchas personas. Esto puede llegar a provocar sentimientos de soledad, frustración, depresión y baja autoestima.

Hoy en día afortunadamente una gran parte de la población, dispone de teléfonos inteligentes que, gracias al gran avance tecnológico en los últimos años, tienen capacidades suficientes para ejecutar

aplicaciones complejas, las cuales pueden ser aliadas para las personas con discapacidad, permitiéndoles normalizar muchos aspectos de su vida. (Fundación ADECCO, 2017).

Aparte de los teléfonos inteligentes, en los últimos años, en el área de Inteligencia Artificial se han desarrollado sistemas inteligentes para mejorar situaciones de la vida diaria, debido a que estos sistemas se adaptan a la situación o al entorno.

La visión artificial o visión por computador es una rama de la Inteligencia Artificial que tiene como finalidad permitir a las "máquinas" ver, extraer información de las imágenes digitales, resolver alguna tarea o entender la escena que están visionando (Maduell, 2015). Otra de las ramas de la Inteligencia Artificial son las Redes Neuronales Artificiales que ha alcanzado trascendental importancia en aplicaciones como el reconocimiento de patrones complejos, detección y reconocimiento de imágenes.

La presente tesis de grado tiene como finalidad aplicar técnicas de Visión Artificial y plantear un modelo de Red Neuronal Artificial para el reconocimiento y la interpretación del alfabeto dactilológico de la Lengua de Señas al lenguaje escrito y oral, implementado en teléfonos inteligentes con sistema operativo Android, para que las personas sordas puedan comunicarse con el resto de la población que no conoce de Lengua de Señas.

1.2 ANTECEDENTES

Desde los inicios de la informática, se ha perseguido la replicación del comportamiento humano en las máquinas. El cuerpo humano ha sido y es considerado como el modelo más eficiente a seguir para determinadas disciplinas, como, por ejemplo: la robótica, la visión artificial, el aprendizaje basado en la experiencia, y muchas otras disciplinas.

En el año 1936 Alan Turing fue el primero en estudiar el cerebro como una forma de ver el mundo de la computación. Sin embargo, los primeros teóricos que concibieron los fundamentos de la computación neuronal fueron Warren McCulloch un neurofisiólogo, y Walter Pitts un matemático, quienes, en 1943 lanzaron una teoría acerca de la forma de trabajar de las neuronas y modelaron una red neuronal simple mediante circuitos eléctricos.

En 1957 Frank Rosenblatt comenzó el desarrollo del Perceptrón. Esta es la red neuronal más antigua, utilizándose hoy en día para aplicación como reconocedor de patrones. Este modelo era capaz de generalizar, es decir, después de haber aprendido una serie de patrones podía reconocer otros similares, aunque no se le hubiesen presentado anteriormente. Sin embargo, tenía una serie de limitaciones, por ejemplo, su incapacidad para resolver el problema de la función OR - exclusiva y, en general, era incapaz de clasificar clases no separables linealmente. En 1959, escribió el libro Principios de Neurodinámica, en el que confirmó que, bajo ciertas condiciones, el aprendizaje del Perceptrón convergía hacia un estado finito (Teorema de Convergencia del Perceptrón)

En 1960 Bernard Widrow y Marcial Hoff desarrollaron el modelo Adaline (ADaptive LINear Elements). Esta fue la primera red neuronal aplicada a un problema real (filtros adaptativos para eliminar ecos en las líneas telefónicas) que se ha utilizado comercialmente durante varias décadas.

A principios de los años 60's, surge la visión artificial o visión por computador con la idea básica de conectar una cámara de video a un computador, esto implicó no solo la captura de imágenes a través de la cámara sino también la comprensión de lo que estas imágenes representaban. Un resultado muy importante de este trabajo y que marcó el inicio de la visión artificial, fue un trabajo de Larry Roberts, el creador de ARPAnet. En 1961 creó un programa, el "mundo de microbloques", en el que un robot podía "ver" una estructura de bloques sobre una mesa, analizar su contenido y reproducirla desde otra perspectiva, demostrando así que esa información visual que había sido mandada al ordenador por una cámara, había sido procesada adecuadamente por él.

Estas técnicas tuvieron un renacimiento en 1964 con el procesamiento de las imágenes recibidas del Ranger 7 que transmitía cerca de la luna. Este procesamiento era básicamente la corrección de distintos tipos de distorsión producida por las cámaras de TV.

En el año 1967 Stephen Grossberg a partir de sus conocimientos fisiológicos realizó una red neuronal artificial: Avalanche, que consistía en elementos discretos con actividad que varía en el tiempo que satisface ecuaciones diferenciales continuas, para resolver actividades como reconocimiento continuo de habla y aprendizaje de los brazos de un robot.

En 1969 surgieron críticas que frenaron, hasta 1982, el crecimiento que estaban experimentando las investigaciones sobre redes neuronales. Marvin Minsky y Seymour Papert, del Instituto

Tecnológico de Massachussets (MIT), publicaron un libro “Perceptrons”. Probaron (matemáticamente) que el Perceptrón no era capaz de resolver problemas relativamente fáciles, tales como el aprendizaje de una función no-lineal. Esto demostró que el Perceptrón era muy débil, dado que las funciones no-lineales son extensamente empleadas en computación y en los problemas del mundo real. A pesar del libro, algunos investigadores continuaron su trabajo. Tal fue el caso de James Anderson, que desarrolló un modelo lineal, llamado Asociador Lineal, que consistía en unos elementos integradores lineales (neuronas) que sumaban sus entradas. Este modelo se basa en el principio de que las conexiones entre neuronas son reforzadas cada vez que son activadas. Anderson diseñó una potente extensión del Asociador Lineal, llamada Brain State in a Box (BSB).

En los años 70's se empezaron a procesar imágenes digitales de cultivos agrícolas e imágenes meteorológicas, también en el área de la biología para la detección y clasificación automática de células.

Ya en los años 80's se empezó a utilizar el procesamiento de imágenes para efectos especiales en la televisión, en el área de medicina para TAC, Resonancia magnética y ultra sonidos entre otros, y en la industria para el control de calidad.

En 1985 John Hopfield provocó el renacimiento de las redes neuronales con su libro: "Computación neuronal de decisiones en problemas de optimización." En 1986 David Rumelhart y G. Hinton. Redescubrieron el algoritmo de aprendizaje de propagación hacia atrás (Backpropagation). A partir de 1986, el panorama fue alentador con respecto a las investigaciones y el desarrollo de las redes neuronales.

A principios de los años 90's, aparece por primera vez mencionada la técnica de recuperación de imágenes basadas en contenido o CBIR (Content-Based Image Retrieval) en un trabajo de T. Kato (1990), el término CBIR fue utilizado para describir a un sistema que recuperaba imágenes de una base de datos basándose en el color y la forma, pero parece que hasta inicios de la década del 2000 se tiene precisión en el concepto orientado hacia la extracción automática de características, así como a la representación de los datos. Las técnicas CBIR usan características de bajo nivel, en general texturas, color y forma para representar a las imágenes.

El gran reto actualmente es el reconocimiento y la clasificación de imágenes en grandes volúmenes, y el mejor ejemplo de esto es la Internet, que se ha convertido en un lugar de confluencia de diversos tipos de imágenes (rostros, coches, dispositivos electrónicos, mapas, paisajes, flora, fauna, etc.). La visión por computador actualmente comprende tanto la obtención como la caracterización e interpretación de las imágenes. Esto supone algoritmos de muy diversos tipos y complejidades.

Por otro lado, en octubre de 2003, en la localidad de Palo Alto del estado de California (Estados Unidos), Andy Rubin, Rich Miner, Chris White y Nick Sears fundan Android Inc. con el objetivo de desarrollar un sistema operativo para móviles basado en Linux. Ya en 2017 Android superó la cuota de mercado de uso de Internet del sistema operativo mundial con un 37,93%, lo que lo sitúa ligeramente por delante de Windows (37,91%) por primera vez., muy por encima de IOS (StatCounter, 2017).

Los teléfonos inteligentes se acercan año a año al rendimiento necesario para los algoritmos más exigentes en Visión por Computadora (detección de objetos, seguimiento visual, etc.). Así que últimamente se están desarrollando aplicaciones de visión por computador para teléfonos inteligentes. Sin ir más lejos OpenCV (librería que proporciona un marco de trabajo de alto nivel para el desarrollo de aplicaciones de visión por computador en tiempo real) desde la versión 2.2 compila y se puede utilizar en Android.

Ya en diciembre de 2017 Google lanza la versión Android Oreo 8.1. Lo más destacado de esta versión es una nueva API de redes neuronales, que aporta inferencia acelerada por hardware el teléfono para ejecutar rápidamente modelos de aprendizaje automático previamente entrenados. Esto puede ser útil cuando se trata de permitir que las aplicaciones en su teléfono hagan cosas como clasificar imágenes o aprender de cómo sus hábitos predicen el comportamiento. Google dijo que diseñaron la API de Neural Networks como una “capa fundamental” para frameworks como TensorFlow Lite, Caffe2 y otros.

A nivel mundial se están desarrollando iniciativas o procesos de investigación sobre las necesidades de las personas con alguna discapacidad, la tecnología móvil se encuentra inmersa en todas partes y se convierte en una herramienta de gran interés para lograr la inclusión e integración de este tipo de poblaciones dentro de la sociedad, solucionando de alguna manera sus dificultades de comunicación, sensitiva, de movilidad, entre otras (Bernal, Salamanca y Cañon, 2003).

En Bolivia, según los datos del el Instituto Nacional de Estadística (INE, 2012), hay 50.562 personas sordas, de las cuales 32.321 tienen "dificultad permanente en hablar, comunicarse o conversar".

En los últimos años se tiene diferentes iniciativas desde la comunidad Sorda y desde el Estado para poder optimizar sus condiciones de Vida. La Población Sorda está organizada mediante la Federación Boliviana de Sordos (FEBOS) la cual ha institucionalizado la celebración de La semana internacional de la persona Sorda, que se celebra la última semana de Septiembre (22-28), se desarrollan actividades artísticas, festivales, ferias y actos protocolares, convocados desde la comunidad Sorda.

Uno de los logros más importantes para la comunidad Sorda es que el 2009, después de un proceso de incidencia desde el 2006, se promulga el decreto supremo 0328, que reconoce la LSB como medio de comunicación de la comunidad sorda del país y señala la necesidad de articular su utilización en diferentes espacios como actos públicos de relevancia nacional, medios de comunicación televisivos y señala el derecho a la educación en LSB. A partir de este decreto se crea el Consejo de la LSB constituido por representantes de la FEBOS y representantes del Estado quien se hace responsable, entre otras cosas, del proceso de Certificación de Competencias a quienes utilizan la LSB (Secretaría Técnica del Comité Nacional Contra el Racismo y toda forma de Discriminación, 2014).

Desde el estado también ha habido respuestas importantes, como el acceso de algunos estudiantes Sordos a las Escuelas Superiores de Formación de Maestros en el país, sin el examen de competencias y gestionando poco a poco la posibilidad de intérpretes de LSB que acompañen dicho proceso de formación.

A continuación, se muestra algunos referentes sobre aplicaciones móviles que tienen como principal objetivo favorecer la comunicación entre una persona sorda y el resto de la población.

- Lengua de Signos textoSIGN, es una herramienta que permite la conversión de texto a Lengua de Signos Española LSE. Este software es un diccionario con un avatar en 3D y más de 1500 palabras. Está disponible la versión Lite de manera gratuita en Android (Signlab, 2013).

- Signslator, es un traductor online, que traduce texto en Castellano a Lengua de signos, Signslator es un microsite, con una versión App, disponible de manera gratuita en Android. Es una herramienta para que oyentes y no oyentes puedan comunicarse, difundiendo y enseñando la lengua de signos (TBWA España, 2014).
- Dilo en señas es un juego para aprender la Lengua de Señas Mexicana LSM, diseñado especialmente para niños sordos. La primera versión de Dilo en señas tiene 89 señas en 7 categorías: familia, alimentos, juguetes, animales, colores, números y abecedario. Y como el juego trata de relacionar los videos con las imágenes, también los niños que no saben leer pueden aprender a decir en señas palabras como: mama, galleta, perro, pelota, etc. (Jaguar Labs, 2015).
- Leesa, es una aplicación (app) reconocida como el mejor proyecto de Bolivia en 2016 por la empresa Tigo y la organización Fundación Trabajo Empresa en Santa Cruz. desarrollada por Óscar Rojas, la aplicación ayuda a las personas que tienen problemas de audición y habla a interrelacionarse de manera efectiva con otros. Esta app le permite al usuario traducir cualquier texto o voz a lenguaje de señas para que la persona que no escucha pueda entenderle. Es de descarga gratuita y está disponible en el Play Store de Android (La Razón, 2017).

En la Carrera de Informática de la Facultad de Ciencias Puras y Naturales de la Universidad Mayor de San Andrés se encuentran pocas investigaciones que contemplen este tipo de temas las cuales podemos mencionar:

- Redes Neuronales para la interpretación del lenguaje de señas, propone desarrollar un prototipo que realice la interpretación del lenguaje de señas al lenguaje oral aplicando las redes neuronales y el tratamiento de imágenes digitales en el reconocimiento de caracteres (Argollo, 2013).
- Tutor inteligente para la enseñanza de la lengua de señas boliviana nivel básico, propone coadyuvar el proceso de enseñanza-aprendizaje de la Lengua de Señas Boliviana (nivel básico) para el buen uso de este especialmente por parte de maestros, maestras y participantes que quieran aprender la LSB destinado a constituir una educación inclusiva, la cual pretende mostrar por medios de un modelo de sistema tutor inteligente y aplicaciones informáticas que serán futuros docentes (Chacolla, 2013).

- Tutor inteligente para la enseñanza de la lectura y escritura para niños sordos. Propone desarrollar un tutor inteligente con parámetros pedagógicos y didácticos, para apoyar y estimular el aprendizaje de la lectura y escritura orientada a niños y niñas de 3 a 6 años con discapacidad auditiva (Cruz, 2016).

1.3 PLANTEAMIENTO DEL PROBLEMA

Los procesos comunicativos son de vital importancia para el desarrollo de cualquier individuo y de una sociedad, estos permiten manifestar ideas, sentimientos, necesidades entre otras emociones. La lengua de señas ayuda a las personas sordas a llevar una vida con comunicación, ya que es el medio con el que podemos comunicarnos personas oyentes y no oyentes. Este lenguaje que podría incluir en mayor medida a las personas con sordera o invalidez del habla, es conocido por un número muy reducido de individuos sin estas discapacidades de nuestra sociedad, lo que hace difícil la comunicación con el resto de la población que no conocen de la lengua de señas, especialmente cuando se encuentran en medio de muchas personas. Esto puede llegar a provocar sentimientos de soledad, frustración, depresión y baja autoestima en la persona que padece de déficit de audición.

1.3.1 PROBLEMA CENTRAL

¿Cómo se puede reconocer e interpretar el Alfabeto Dactilológico de la Lengua de Señas Boliviana en dispositivos móviles?

1.3.2 PROBLEMAS SECUNDARIOS

- Deficiente comunicación de personas sordas con el resto de la población, debido a la dificultad en el aprendizaje de la Lengua de Señas los cual genera exclusión social, baja autoestima y sentimientos de soledad a las personas sordas.
- Los sistemas de reconocimiento de imágenes requieren de una gran cantidad de datos, tiempo y una gran capacidad de procesamiento por parte del hardware, lo que requiere de un alto costo de inversión en equipamiento, esto limita el desarrollo de este tipo de sistemas.
- Bajo rendimiento de sistemas de visión artificial en dispositivos móviles de gama baja, lo cual limita el acceso de estas aplicaciones a personas con bajos recursos económicos.

- Sistemas de visión artificial inaccesibles a cualquier persona debido a la necesidad de adquirir algún dispositivo especializado.
- Resultados en un tiempo de traducción no razonables en dispositivos móviles, debido a la necesidad de conexión a un servidor remoto a través de internet.
- Deficiente comprensión y difusión del alfabeto dactilológico de la lengua de señas a la población oyente.

1.4 DEFINICIÓN DE OBJETIVOS

1.4.1 OBJETIVO GENERAL

Desarrollar una aplicación móvil para el reconocimiento e interpretación del alfabeto dactilológico de la lengua de señas, basado en redes neuronales artificiales.

1.4.2 OBJETIVOS ESPECÍFICOS

- Representar las imágenes de señas capturadas en su correspondiente traducción al lenguaje oral, indicando el nivel de confianza de que la imagen represente a dicha letra del alfabeto.
- Utilizar la técnica de transferencia de aprendizaje de redes neuronales artificiales, la cual permite reducir de manera considerable los requerimientos de hardware, datos y tiempo que este tipo de sistemas normalmente demandan.
- Optimizar técnicas y algoritmos de visión artificial y redes neuronales artificiales para su funcionamiento eficiente en dispositivos móviles de gama baja y accesible a personas con bajos recursos económicos.
- Desarrollar una aplicación para dispositivos móviles accesible, para cualquier persona pueda hacer uso de ella sin necesidad de adquirir algún dispositivo especializado.
- Desarrollar una aplicación eficiente para dispositivos móviles, que sea capaz de mostrar resultados en un tiempo razonable, utilizando los recursos físicos del dispositivo móvil, de tal manera que no sea necesario la conexión a un servidor remoto a través de internet.
- Facilitar la comprensión y difundir el alfabeto dactilológico de la lengua de señas para la población oyente.

1.5 HIPÓTESIS

El uso de redes neuronales artificiales permite realizar el reconocimiento e interpretación del alfabeto dactilológico de la lengua de señas en dispositivos móviles, con un grado de confiabilidad de al menos 90%

1.6 JUSTIFICACIÓN

1.6.1 JUSTIFICACIÓN ECONÓMICA

La presente tesis de grado de sistema de reconocimiento e interpretación del alfabeto dactilológico para dispositivos móviles se justifica económicamente porque permitirá eliminar el costo que existiría pagar a un profesional traductor de la lengua de señas al lenguaje oral. Además, el presente trabajo de grado se desarrollará bajo plataforma de software libre y tecnologías open source, lo cual permitirá desarrollar el proyecto para que esté disponible de forma gratuita para los usuarios finales, de tal forma que el uso del sistema no debe generar gastos significativos adicionales para el usuario.

1.6.2 JUSTIFICACIÓN SOCIAL

La presente tesis de grado de sistema de reconocimiento e interpretación del alfabeto dactilológico para dispositivos móviles es un trabajo de carácter social, ya que permitirá mejorar la integración y la calidad de vida de las personas con deficiencia auditiva, como también promover y facilitar el aprendizaje de la lengua de señas a personas oyentes.

1.6.3 JUSTIFICACIÓN CIENTÍFICA

En la presente tesis de grado se requiere el uso de Redes Neuronales Artificiales para el reconocimiento de la lengua de señas. Las redes neuronales actualmente han tenido importantes avances científicos y aplicaciones como el reconocimiento de patrones, reconocimiento del habla y del lenguaje escrito, análisis del mercado de valores, clasificación de secuencias de ADN, juegos y muchas otras más.

Hoy en día la tecnología móvil se encuentra inmersa en todas partes y se ha convertido en una herramienta de gran interés para lograr la inclusión e integración de este tipo de poblaciones dentro de la sociedad, solucionando de alguna manera sus dificultades de comunicación.

1.7 ALCANCES Y LÍMITES

1.7.1 ALCANCES

- En la presente tesis de grado se realizará un sistema basado en la captura de imágenes de una seña del alfabeto dactilológico a través de la cámara de un dispositivo móvil con sistema operativo Android.
- Las imágenes serán procesadas por el sistema el cual realizará el reconocimiento de la seña mediante redes neuronales artificiales.
- El procesamiento de las imágenes se realizará utilizando los recursos físicos del dispositivo móvil de modo que no será necesario utilizar un servidor remoto el cual necesariamente tendría que ser contactado por medio de una conexión a internet lo cual generaría gastos adicionales al usuario.
- Se realizará la traducción al lenguaje oral mediante el altavoz del dispositivo móvil.

1.7.2 LÍMITES

- El sistema solo realizara el reconocimiento e interpretación del alfabeto dactilológico con la exclusión de las letras “j”, “ñ” y “z” ya que estas para su representación necesitan del movimiento de las manos.
- La aplicación móvil solo estará disponible para dispositivos móviles con sistema operativo Android con versión superior a 4.0.3 (IceCreamSandwich)
- El sistema no puede reconocer e interpretar otro tipo de señas con movimiento de la lengua de señas.
- El sistema solo podrá realizar la interpretación de una seña del alfabeto dactilológico de la lengua de señas y traducirlo al lenguaje oral, no realizara lo contrario es decir la interpretación del lenguaje oral a la lengua de señas.
- El sistema reconocerá e interpretará una seña a la vez con una iluminación y distancia adecuada.

1.8 APORTES

1.8.1 PRÁCTICO

- Una herramienta útil para la traducción de la lengua de señas para personas que no conocen dicho lenguaje.
- Una herramienta útil para difundir el alfabeto dactilológico.
- Reducción de costos en la contratación de un traductor profesional.
- Inclusión de personas con deficiencia auditiva en la comunidad.

1.8.2 TEÓRICO

- Una herramienta para posteriores investigaciones en el área de traducción de lenguajes.
- Profundización en la investigación de redes neuronales artificiales y tecnología móvil para su uso en reconocimiento de imágenes.



CAPÍTULO II

MARCO TEÓRICO

2.1 INTELIGENCIA ARTIFICIAL

La Inteligencia Artificial o IA es una rama de las ciencias computacionales encargada de estudiar modelos de cómputo capaces de realizar actividades propias de los seres humanos en base a dos de sus características primordiales: El razonamiento y la conducta. Existen distintas definiciones de IA de acuerdo a cuatro enfoques que se muestran a continuación:

- Sistemas que actúan como humanos
IA según Elaine Rich (1991) Es el estudio de cómo hacer que las computadoras hagan cosas que por el momento las personas las realizan de mejor manera
- Sistemas que piensan como humanos.
IA según Haugeland (1985) Es el esfuerzo de hacer que los ordenadores piensen... *máquinas con mentes* en el más amplio sentido literal.
- Sistemas que piensan racionalmente
IA según Charniak y McDermott (1985) Es el estudio de las facultades mentales a través del estudio de modelos computacionales.
- Sistemas que actúan racionalmente
IA según Shalkoff (1990) Es el estudio que busca explicar y emular el comportamiento inteligente en términos de procesos computacionales

La IA es muy interdisciplinaria, y en ella intervienen disciplinas tan variadas como la Neurociencia, la Psicología, las Tecnologías de la Información, la Ciencia Cognitiva, la Física, las Matemáticas, etc.

Hoy en día la IA está principalmente relacionada con actividades en las siguientes áreas de investigación: Redes Neuronales Artificiales, Algoritmos Genéticos, Realidad Virtual, Vida Artificial, Sistemas Expertos, Lógica Difusa, Visión artificial o Visión por Computador y Reconocimiento del habla.

En los últimos años ha habido un desarrollo explosivo de la investigación básica y aplicada alrededor de las Redes Neuronales Artificiales (denominadas habitualmente como RNA o en inglés como ANN). Entre las razones de este desarrollo explosivo, destaca el que las Redes Neuronales Artificiales presentan en ciertos campos claras ventajas sobre las computadoras digitales comunes, llamadas del tipo Von Neumann, (Von Neumann, 1945) pues no sólo son capaces de aprender de la experiencia sin grandes complicaciones de software o de hardware, sino que pueden resolver en forma sencilla y económica algunos de los problemas que constituyen los mayores retos para las computadoras tradicionales, tales como el reconocimiento de patrones complejos, procesamiento de imágenes, generación e interpretación de lenguaje natural y problemas de clasificación y diagnóstico difusos. (Lara, 2015)

2.1.1 EL ASCENSO DE MACHINE LEARNING

Dentro de Inteligencia Artificial podemos clasificar dos grupos fundamentales: aplicada o general. La aplicada es mucho más común: los sistemas diseñados para negociar inteligentemente acciones o maniobrar un vehículo autónomo caerían en esta categoría. Los generalizados son sistemas o dispositivos que en teoría pueden manejar cualquier tarea, son menos comunes, pero aquí es donde están sucediendo algunos de los avances más emocionantes de hoy. También es el área que ha llevado al desarrollo del Machine Learning en español Aprendizaje Automático (Analytics10, 2018).

Dos importantes avances llevaron a la aparición del Machine Learning como el vehículo que está impulsando el desarrollo de Inteligencia Artificial con la velocidad que tiene actualmente. Uno de ellos fue la realización, acreditada a Arthur Samuel en 1959, de que en lugar de enseñar a las computadoras todo lo que necesitan saber sobre el mundo y cómo llevar a cabo las tareas, podría ser posible enseñarles a aprender por sí mismos. El segundo, más recientemente, fue la aparición de Internet, y el enorme aumento en la cantidad de información digital que se genera, almacena y pone a disposición para el análisis (Analytics10, 2018).

Una vez que estas innovaciones estaban en su lugar, los ingenieros se dieron cuenta de que en lugar de enseñar a las computadoras y máquinas cómo hacer todo, sería mucho más eficiente enseñarles a pensar como seres humanos, y luego conectarlos a Internet para darles acceso a toda la información en el mundo (Analytics10, 2018).

2.2 REDES NEURONALES ARTIFICIALES

Uno de los temas de la IA que ha alcanzado trascendental importancia son las Redes Neuronales Artificiales (RNA) y en dicho tema, una de las aplicaciones más importantes es el reconocimiento de patrones complejos.

Las RNA son sistemas compuestos por estructuras de red con un gran número de conexiones entre diferentes capas de procesadores, los cuales a su vez tienen asignadas diferentes funciones, dentro de dichos procesadores se efectúa una labor de aprendizaje por la reproducción de las salidas de un conjunto de señales de entrenamiento. (Vázquez, 2008).

Por otro lado, según (Zampayo, 2004) las Redes Neuronales Artificiales son sistemas de cómputo distribuidos y paralelos inspirados en la estructura del cerebro humano. El cerebro humano consta de miles de millones de neuronas; cada una conectada a miles de otras neuronas en una estructura distribuida, con paralelismo masivo. Este tipo de estructura otorga al cerebro una gran ventaja en la mayoría de las capacidades perceptivas, motrices y creativas.

Se puede decir que las redes neuronales constituyen una tecnología, la cual trata con éxito algunos problemas clásicos de la IA, haciendo énfasis en el reconocimiento de formas y de la palabra hablada.

2.2.1 FUNDAMENTOS BIOLÓGICOS DE LAS REDES NEURONALES NATURALES

Una neurona biológica es una célula especializada en procesar información. Está compuesta por el cuerpo de la célula (soma) y dos tipos de ramificaciones: el axón y las dendritas. La neurona recibe las señales (impulsos) de otras neuronas a través de sus dendritas y transmite señales generadas por el cuerpo de la célula a través del axón. La figura 2.1 muestra los elementos de una red neuronal natural. (Ponce, 2010)

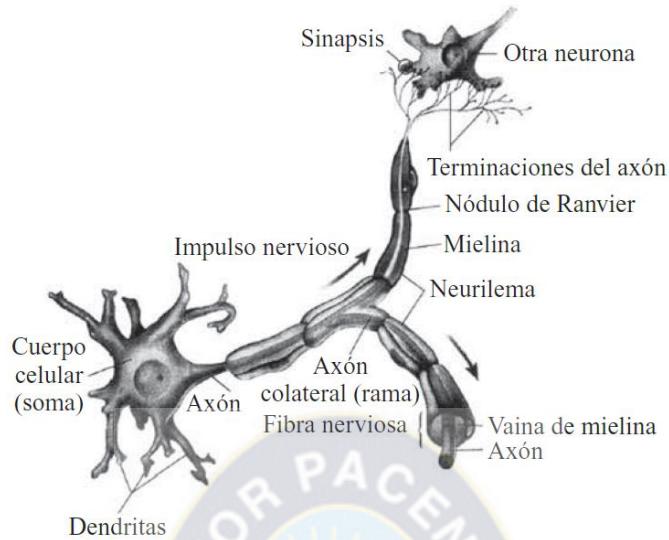


Figura 2.1: Elementos neurales

Fuente: (Ponce, 2010)

En general, una neurona consta de un cuerpo celular más o menos esférico, de 5 a 10 micras de diámetro, del que salen una rama principal, el axón y varias ramas más cortas que corresponden a las dendritas. (Ponce, 2010)

Una de las características de las neuronas es su capacidad de comunicarse. En forma concreta, las dendritas y el cuerpo celular reciben señales de entrada; el cuerpo celular las combina e integra y emite señales de salida. El axón transmite dichas señales a las terminales axónicas, los cuales distribuyen la información. Se calcula que en el cerebro humano existen aproximadamente 1015 conexiones. (Ponce, 2010)

Las señales que se utilizan son de dos tipos: eléctricas y químicas. La señal generada por la neurona y transportada a lo largo del axón es un impulso eléctrico, mientras que la señal que se transmite entre los terminales axónicos de una neurona y las dendritas de la otra es de origen químico. (Ponce, 2010)

Para establecer una similitud directa entre la actividad sináptica y la analogía con las redes neurales artificiales podemos considerar que las señales que llegan a la sinapsis son las entradas a la neurona; éstas son ponderadas (atenuadas o simplificadas) a través de un parámetro denominado peso, asociado a la sinapsis correspondiente. Estas señales de entrada pueden excitar a la neurona (sinapsis con peso positivo) o inhibirla (peso negativo). (Ponce, 2010)

El efecto es la suma de las entradas ponderadas. Si la suma es igual o mayor que el umbral de la neurona, entonces la neurona se activa (da salida). Esta es una situación de todo o nada; cada neurona se activa o no se activa. La facilidad de transmisión de señales se altera mediante la actividad del sistema nervioso. Las sinapsis son susceptibles a la fatiga, deficiencia de oxígeno y la presencia de anestésicos, entre otros. Esta habilidad de ajustar señales es un mecanismo de aprendizaje. (Ponce, 2010)

2.2.2 ELEMENTOS FUNCIONALES DE LA NEURONA ARTIFICIAL

Un esquema básico de una RNA se observa en la figura 2.2, la cual presenta las diferentes capas que contiene esta topología, que es una estructura que se conoce con el nombre de feed-forward (hacia adelante) debido al flujo de la información.

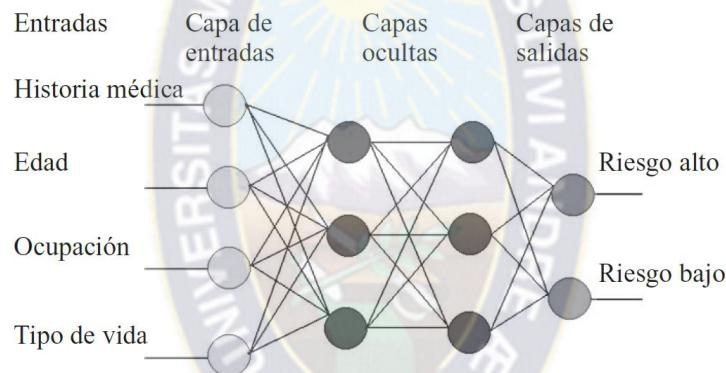


Figura 2.2: Esquema básico de una RNA.

Fuente: (Ponce, 2010).

La neurona artificial es una unidad procesadora con cuatro elementos funcionales (Lara, 2012), como se detalla en la figura 2.3.

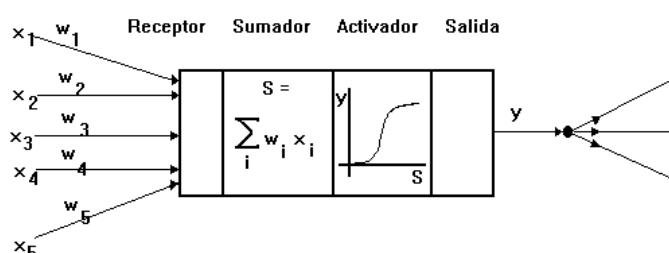


Figura 2.3: Esquema de una neurona artificial.

Fuente: (Lara, 2012).

- **El elemento receptor**, a donde llegan una o varias señales de entrada x_i , que generalmente provienen de otras neuronas y que son atenuadas o amplificadas cada una de ellas con arreglo a un factor de peso w_i que constituye la conectividad entre la neurona fuente de donde provienen y la neurona de destino en cuestión.
- **El elemento sumador**, que efectúa la suma algebraica ponderada de las señales de entrada, ponderándolas de acuerdo con su peso, aplicando la siguiente expresión:

$$S = \sum w_i x_i$$

- **El elemento de función de activación**, que aplica una función no lineal de umbral (que frecuentemente es una función escalón o una curva logística) a la salida del sumador para decidir si la neurona se activa, disparando una salida o no.
- **El elemento de salida** que es el que produce la señal, de acuerdo con el elemento anterior, que constituye la salida de la neurona.

En muchas redes las unidades de proceso tienen respuesta de la forma (Ponce, 2010):

$$y = f\left(\sum_k w_k x_k\right)$$

donde:

x_k : señales de salida de otros nodos o entradas externas.

w_k : pesos de las ligas de conexión.

$f(\cdot)$: función no lineal simple.

La función f puede ser sigmoidal, tangente hiperbólica, escalón, entre otras. Cada unidad de proceso tiene una tarea simple: recibe la entrada de otras unidades o de fuentes externas y procesa la información para obtener una salida que se propaga a otras unidades (Ponce, 2010).

Una red puede tener una estructura arbitraria, pero las capas que contienen estas estructuras están definidas de acuerdo con su ubicación en la topología de la red neuronal. Las entradas externas son aplicadas en la primera capa, y las salidas se consideran la última capa. Las capas internas que

no se observan como entradas o salidas se denominan capas ocultas. Por convención, las entradas no se consideran como capa porque no realizan procesamiento (Ponce, 2010).

La entrada total u de una unidad k es la suma de los pesos de las entradas conectadas, más un bias θ :

$$u = \sum_j w_j x_j + \theta$$

Este modelo neuronal es el utilizado en casi todas las Redes Neuronales artificiales, variando únicamente el tipo de función de activación.

2.2.3 FUNCIONES DE ACTIVACIÓN

Algunas de las funciones de activación más usadas son las siguientes:

2.2.3.1 FUNCIÓN ESCALÓN

La función de activación escalón se asocia a neuronas binarias en las cuales, cuando la suma de las entradas es mayor o igual que el umbral de la neurona. La activación es 1; si es menor, la activación es 0 (o -1). (Ponce, 2010)

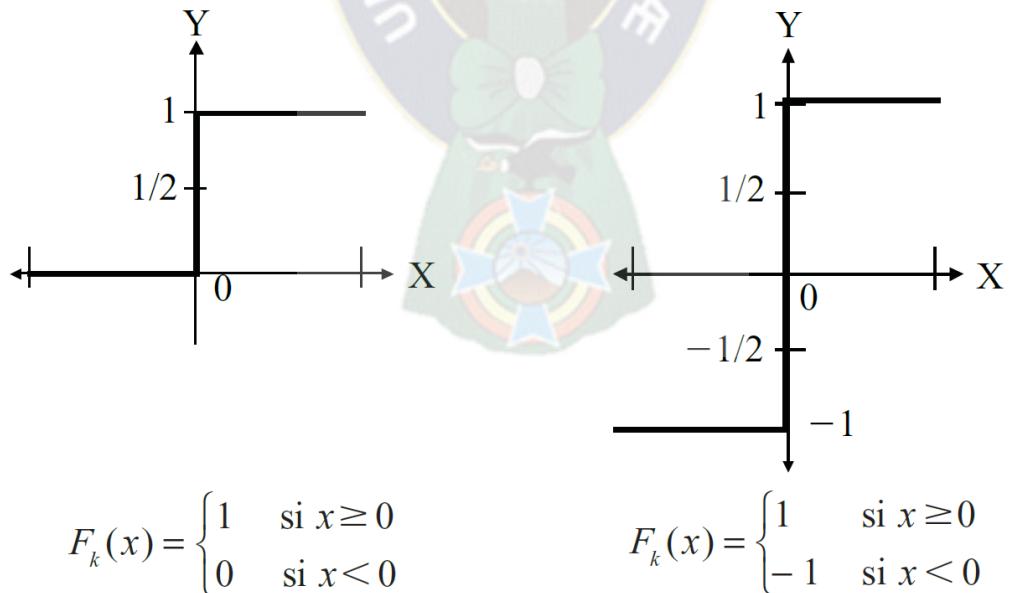


Figura 2.4: Función de activación escalón.

Fuente: (Ponce, 2010)

2.2.3.2 FUNCIÓN LINEAL Y MIXTA

La función lineal o identidad responde a la expresión $F_k(u) = u$. En las neuronas con función mixta, si la suma de las señales de entrada es menor que un límite inferior, la función se define como 0 (o -1). Si dicha suma es mayor o igual que el límite superior, entonces la activación es 1. Si la suma de entrada está comprendida entre los dos límites, superior e inferior, entonces la activación se define como una función lineal de la suma de las señales de entrada, (Ponce, 2010) véase la figura 2.5.

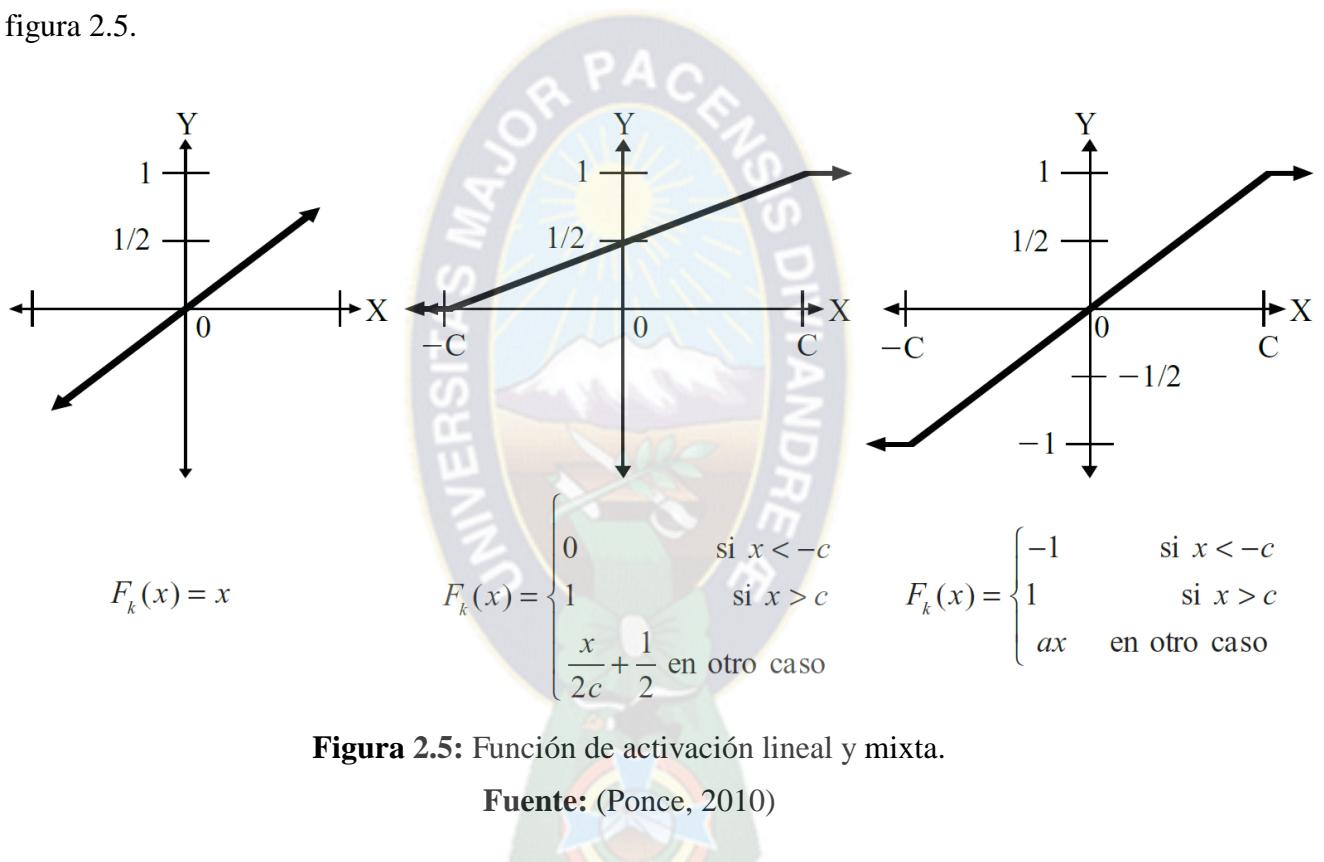


Figura 2.5: Función de activación lineal y mixta.

Fuente: (Ponce, 2010)

2.2.3.3 FUNCIÓN TANGENTE HIPERBÓLICA

La función de activación tangente hiperbólica se emplea en los casos que presentan variaciones suaves de valores positivos y negativos de la señal a clasificar. Como se puede ver en su descripción en la figura 2.6, es una de las funciones más empleadas en entrenamientos supervisados, como en el caso del entrenamiento de retropropagación del error. (Ponce, 2010)

Debe tenerse cuidado de emplear esta figura entre los umbrales positivos y negativos antes de la saturación, de otra forma la salida siempre generará valores saturados iguales a 1 y -1.

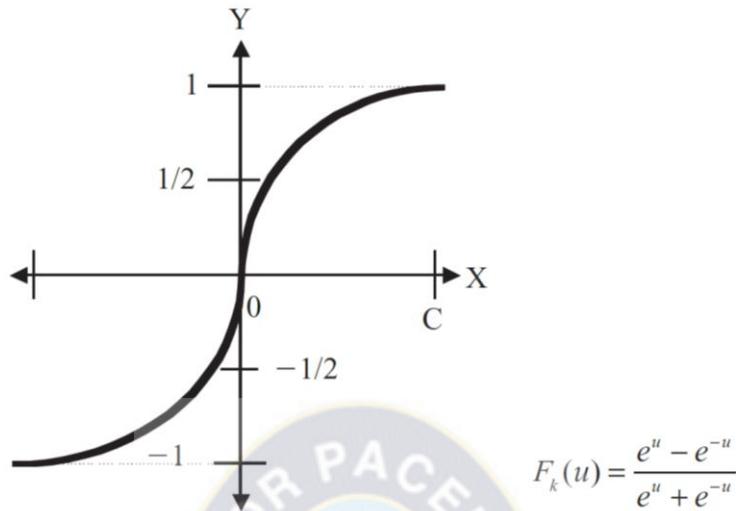


Figura 2.6: Función tangente hiperbólica

Fuente: (Ponce, 2010)

2.2.3.4 FUNCIÓN SIGMOIDAL

Con la función sigmoidal el valor dado por la función es cercano a uno de los valores asintóticos. Esto hace que, en la mayoría de los casos, el valor de salida esté comprendido en la zona alta o baja del sigmoide. De hecho, cuando la pendiente es elevada, esta función tiende a la función escalón. Sin embargo, la importancia de la función sigmoidal es que su derivada siempre es positiva y cercana a cero para los valores grandes positivos o negativos; además, toma su valor máximo cuando $x = 0$. Esto hace que se puedan utilizar reglas de aprendizaje definidas para las funciones escalón, con la ventaja, respecto a esta función, de que la derivada está definida en todo el intervalo, (Ponce, 2010) véase la figura 2.7.

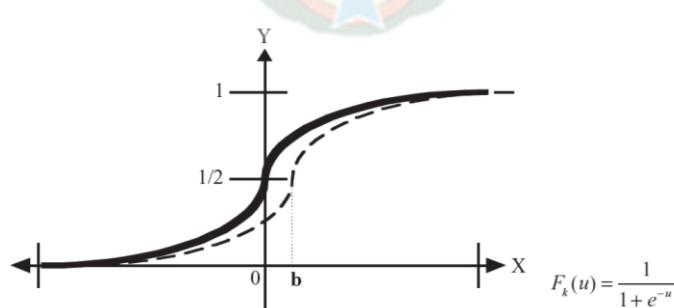


Figura 2.7: Función sigmoidal.

Fuente: (Ponce, 2010)

2.2.3.5 FUNCIÓN DE GAUSS

Los mapeos ocultos algunas veces pueden realizarse con un solo nivel de neuronas mediante el uso de funciones de activación tipo Gauss, en lugar de funciones tipo sigmoidales, (Ponce, 2010) véase la figura 2.8.

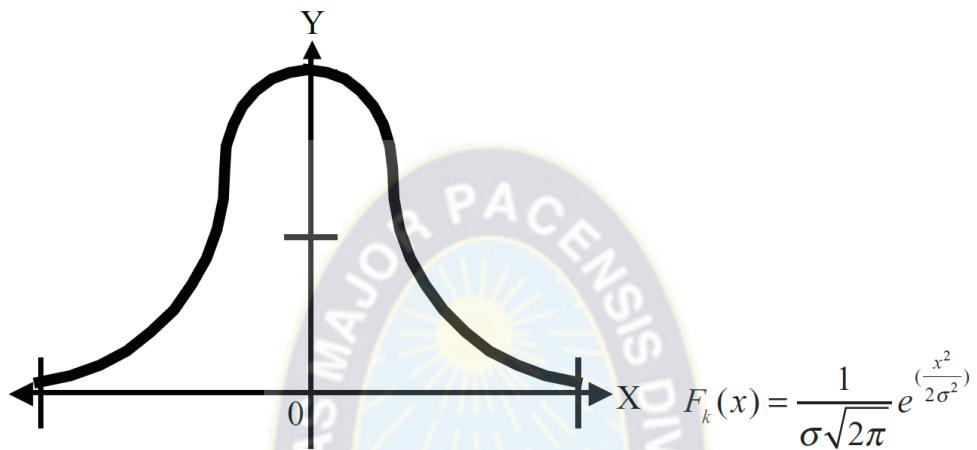


Figura 2.8: Función de activación gaussiana

Fuente: (Ponce, 2010)

2.2.4 APRENDIZAJE

El aprendizaje es la clave de la plasticidad de una RNA y esencialmente es el proceso en el que se adaptan las sinapsis, para que la RNA responda de un modo distinto a los estímulos del medio. En una RNA toda la información adquirida se guarda en el valor de cada peso sináptico. Existen dos tipos de aprendizaje, el Aprendizaje Supervisado y No Supervisado.

2.2.5 APRENDIZAJE SUPERVISADO

Necesitan un conjunto de datos de entrada previamente clasificado o cuya respuesta objetivo se conoce. Ejemplo de este tipo de redes son el Perceptrón, la red Adaline, Madaline, y Retropropagación.

2.2.5.1 PERCEPTRÓN

Desarrollado por Rosenblatt (1958), consiste en una neurona procesadora, con sus elementos de entrada, sumador, activador y de salida. A la cual llegan señales de entrada $x_j, j = 1, 2, \dots, n$ cada una con una a través de una línea con conductividad o peso asociado w_i . (Ponce, 2010)

Un perceptrón convencional tiene una función de no linealidad binaria y la topología que muestra la figura 2.9

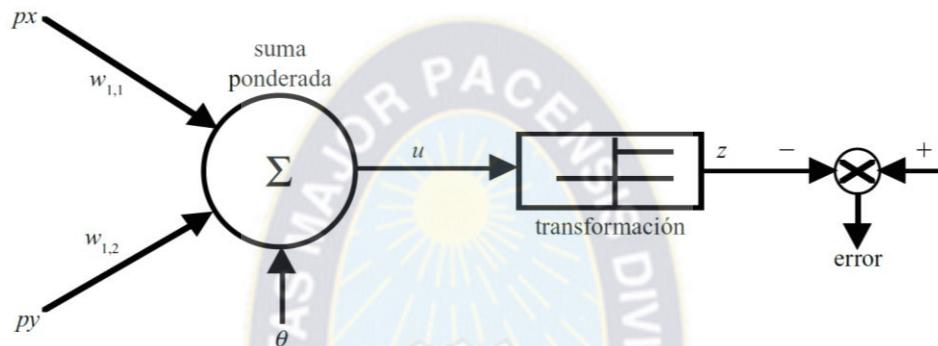


Figura 2.9: Perceptrón clásico propuesto por McCulloch – Pitts.

Fuente: (Ponce, 2010)

El algoritmo de aprendizaje del perceptrón funciona para aprender funciones binarias linealmente separables, ya que de otra manera el algoritmo no convergería ni produciría la mejor solución. Debido a que las salidas son binarias, se emplean unidades lineales de umbral. Cada unidad calcula la suma u con pesos de las N entradas $x_j, j = 1 \dots N$, y genera una salida binaria y (Ponce, 2010):

$$u = \sum_{j=0}^N w_j x_j = \vec{w}^T \vec{x}$$

$$y = \begin{cases} -1, & u \leq 0 \\ +1, & u > 0 \end{cases}$$

La constante de bias es 1, siendo las entradas, salidas y valores objetivo asumidos como ± 1 o cero. Los pesos son actualizados por un número de reglas simples, comparando las salidas $y(\vec{x})$ con los objetivos $t(\vec{x})$. Los pesos son adaptados con la ecuación (Ponce, 2010):

$$\Delta \vec{w} = \begin{cases} 2\eta t \vec{x}, & \text{si } t \neq y \\ 0, & \text{en otros casos} \end{cases}$$

Donde:

η es una constante positiva pequeña que controla la tasa de aprendizaje usualmente entre 0 y 1.

Al mejorar la exactitud de la clasificación, el sistema comete menos errores y los cambios en los pesos se vuelven menos frecuentes. Una tasa efectiva de aprendizaje puede alentar el proceso, por lo que lograr la clasificación perfecta puede tomar un largo tiempo (Ponce, 2010).

La forma de entrenamiento se puede realizar de una manera recursiva empleando la siguiente expresión.

$$\Delta \vec{w} = w(k+1) = w(k) + \eta x(k)e(k)$$

$$\text{donde } e(k) = yd(k) - y(k)$$

Se define el error como $e(k)$, el valor deseado $yd(k)$ y valor actual de salida de la neurona como $y(k)$.

El algoritmo se define como:

- 1) fijar pesos iniciales, con valores aleatorios
- 2) establecer los valores de entradas x_1, x_2, \dots, x_n
- 3) calcular la salida de la neurona

$$y(k) = f(\sum x_i w_i)$$

- 4) Actualizar los pesos

$$\Delta \vec{w} = w(k+1) = w(k) + \eta x(k)e(k)$$

$$\text{donde } e(k) = yd(k) - y(k)$$

- 5) continuar hasta

$$e(k) \leq \xi$$

$$\xi = \text{Tolerancia}$$

En caso de no cumplir con la condición del paso 5, se regresa de nuevo al punto 3.

Un perceptrón de dos entradas puede separar un plano en dos secciones, ya que su ecuación forma una línea recta, como muestra la figura 2.10 (Ponce, 2010).

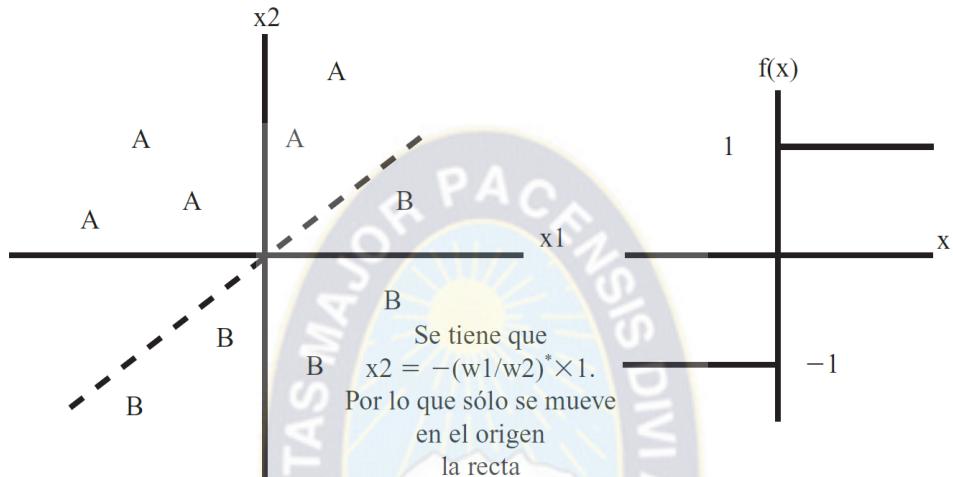


Figura 2.10: Clasificación de patrones con el perceptrón.

Fuente: (Ponce, 2010)

Considerando las entradas x_1 y x_2 , con sus respectivos pesos w_1 y w_2 y un bias θ , se obtiene la ecuación:

$$w_1x_1 + w_2x_2 + \theta = 0$$

Despejando resulta una ecuación donde los pesos determinan la pendiente de la línea y el bias determina el desplazamiento vertical de la clasificación lineal:

$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{\theta}{w_2}$$

Esta recta se denomina frontera de decisión y es ortogonal al vector de los pesos de la red.

2.2.5.2 ADALINE

La red ADALINE (Adaptive Linear Neuron) tiene una topología similar a la del perceptrón, excepto porque emplea una función de activación lineal, usando para su entrenamiento un método de mínimos cuadrados (LMS). (Ponce, 2010)

Un esquema de predicción básico es el siguiente, en el cual se presenta un entrenamiento con datos del sistema tanto de la entrada como de la salida. Después de realizado el entrenamiento se elimina la entrada de datos de la salida del sistema, como lo muestra la figura 2.11.

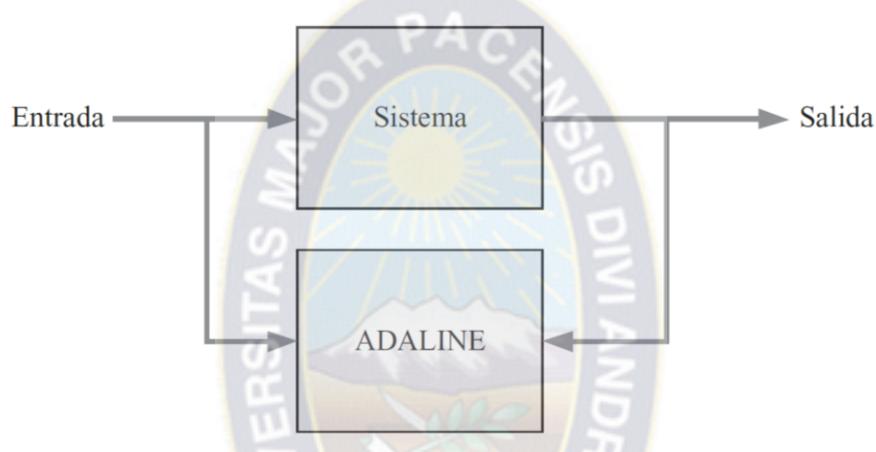


Figura 2.11: Sistema de entrenamiento para una red ADALINE.

Fuente: (Ponce, 2010).

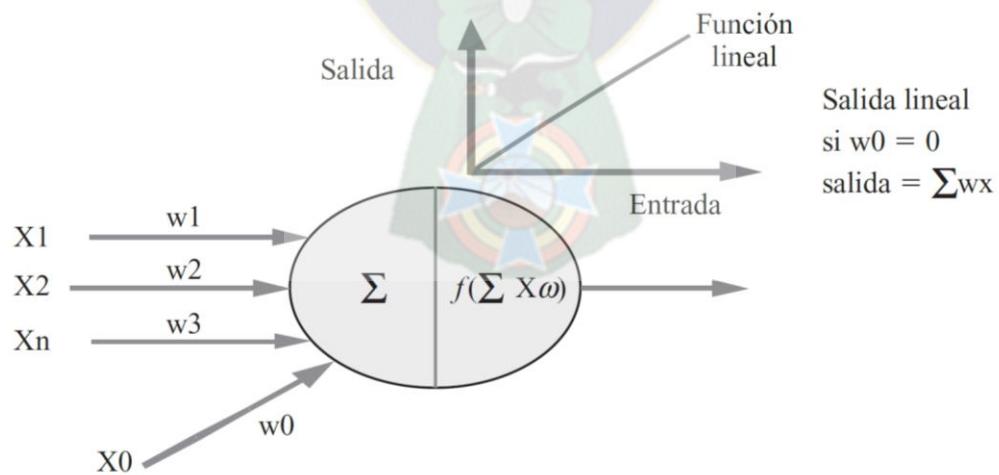


Figura 2.12: ADALINE (con salida de activación lineal).

Fuente: (Ponce, 2010).

Las redes ADALINE son estructuras de una capa con una función de escalón como no linealidad. Adicionalmente, las entradas pueden ser de forma continua, mientras que en el perceptrón son binarias de ± 1 . Los pesos son ajustados mediante la regla de Widrow-Hoff para minimizar la diferencia entre la salida y el objetivo. Este algoritmo es una implementación iterativa de la regresión lineal, reduciendo el error cuadrado de un ajuste lineal (Ponce, 2010).

Los pesos se actualizan de manera general por:

$$\Delta \vec{w} = \eta(t - u) \frac{\vec{x}}{\|\vec{x}\|^2}$$

Empleando el error cuadrático medio se tiene

$$e^2 = \frac{1}{2} \sum_{k=1}^h e_k^2$$

Donde h es el número de patrones de entrada; partiendo de la definición básica del error se tiene $e = (d(k) - s(k))$, calculando la salida por $s(k) = \sum_{j=1}^n x_j w_j$, como se desea reducir el gradiente descendiente se puede expresar a través de $\Delta w_j = -\alpha \left[\frac{\partial e_k^2}{\partial w_i} \right]$

donde α = coeficiente de entrenamiento

$$\Delta w = -\alpha \frac{\partial e_k^2}{\partial s_k} \frac{\partial s_k}{\partial w_i}$$

Si s_k , se tiene una salida lineal, $s(k) = \sum_{j=1}^n x_j w_j$, se puede $\frac{\partial e_k^2}{\partial w_i} = -e_k x_k$

La actualización de los pesos se puede realizar por la siguiente expresión:

$$\Delta w = w(k) + \alpha x(k) e(k)$$

2.2.5.3 MADALINE

La red madaline posee dos capas, una capa de entrada y una capa de salida. La primera capa está formada por neuronas del tipo Adaline. A pesar de contar con dos capas, solamente puede ajustar los pesos de las neuronas Adaline. Los pesos de la capa de salida tienen un valor igual a 1, y cada neurona entrega su valor de +1 o -1 a la capa siguiente.

Las salidas de las neuronas se calculan mediante una regla de mayorías, si más de la mitad de neuronas Adaline tiene una salida de 1, la salida total del sistema será 1. Caso contrario la salida será -1. Los principales usos son filtros y eliminación de ruidos en señales portadoras (Bronson, 2007).

2.2.6 APRENDIZAJE NO SUPERVISADO

En el aprendizaje no supervisado los datos de entrada son ingresados a la red, y los mismos son tomados como valores aleatorios. Se busca similitudes entre los datos y se irán auto organizando. En este tipo de entrenamiento no se necesita especificar a la red un vector de salidas deseadas. Ejemplos de estas redes son las memorias asociativas, redes de Hopfield, máquina de Boltzman y las redes Kohonen.

2.3 REDES NEURONALES CONVOLUCIONALES

Las redes neuronales convolucionales o CNNs por sus siglas en inglés son un tipo especializado de redes neuronales recomendadas para el procesamiento de datos con una topología en forma de malla o grid. El tipo de datos más utilizado con este tipo de redes son las imágenes (mallas de x e y píxeles).

En los últimos años, las redes neuronales profundas han dado lugar a resultados innovadores en una variedad de problemas de reconocimiento de patrones, como la visión por computadora y el reconocimiento de voz. Uno de los componentes esenciales que conducen a estos resultados ha sido un tipo especial de red neuronal llamada red neuronal convolucional (Olah, 2014).

En su forma más básica, las redes neuronales convolucionales pueden considerarse como un tipo de red neuronal que usa muchas copias idénticas de la misma neurona.¹ Esto permite que la red tenga muchas neuronas y exprese modelos computacionalmente grandes, manteniendo el número real de neuronas, parámetros, los valores que describen cómo se comportan las neuronas, que deben aprenderse bastante pequeños (Olah, 2014).

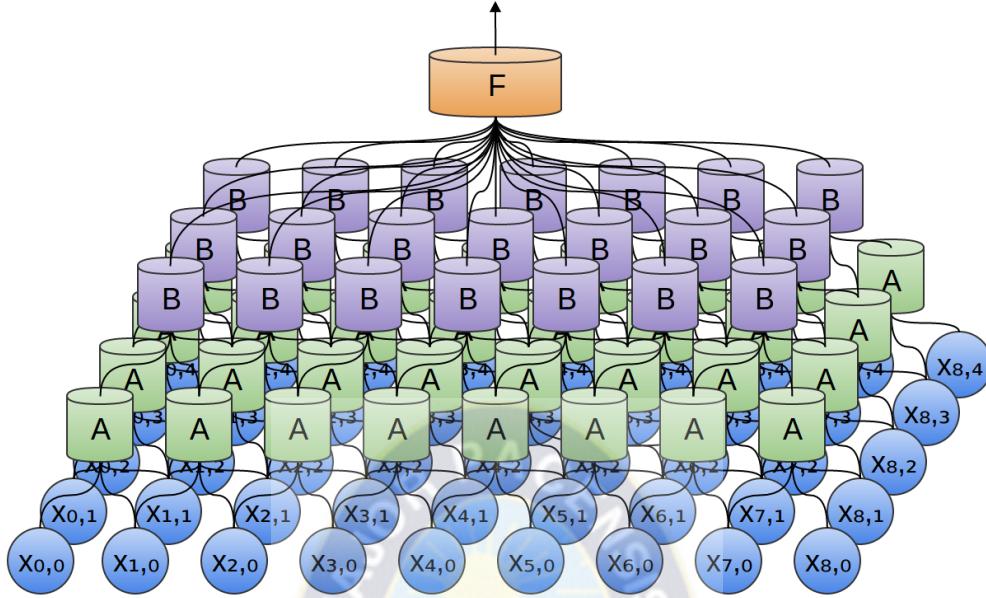


Figura 2.13: Una red neuronal convolucional 2D.

Fuente: (Olah, 2014).

Este truco de tener copias múltiples de la misma neurona es más o menos análogo a la abstracción de funciones en matemáticas e informática. Cuando programamos, escribimos una función una vez y la usamos en muchos lugares; no escribir el mismo código cientos de veces en diferentes lugares hace que sea más rápido programar y resulta en menos errores. Del mismo modo, una red neuronal convolucional puede aprender una neurona una vez y usarla en muchos lugares, lo que facilita el aprendizaje del modelo y la reducción del error.

2.3.1 ESTRUCTURA DE REDES NEURONALES CONVOLUCIONALES

Supongamos que quiere una red neuronal para mirar muestras de audio y predecir si un humano está hablando o no. Tal vez quieras hacer más análisis si alguien está hablando.

Obtienes muestras de audio en diferentes momentos. Las muestras están espaciadas uniformemente.



Figura 2.14: Ejemplo de una red neuronal convolucional 2D paso 1

Fuente: (Olah, 2014).

La forma más sencilla de tratar de clasificarlos con una red neuronal es simplemente conectarlos a una capa totalmente conectada. Hay un montón de neuronas diferentes, y cada entrada se conecta a cada neurona.

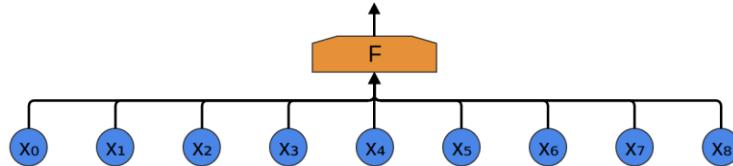


Figura 2.15: Ejemplo de una red neuronal convolucional 2D paso 2

Fuente: (Olah, 2014).

Un enfoque más sofisticado advierte un tipo de simetría en las propiedades que es útil buscar en los datos. Nos preocupan mucho las propiedades locales de los datos: ¿qué frecuencia de sonidos hay en un momento determinado? ¿Están aumentando o disminuyendo? Y así.

Nos importan las mismas propiedades en todos los momentos. Es útil conocer las frecuencias al principio, es útil conocer las frecuencias en el medio, y también es útil conocer las frecuencias al final. Una vez más, tenga en cuenta que estas son propiedades locales, en el sentido de que solo tenemos que mirar una pequeña ventana de la muestra de audio para determinarlas.

Entonces, podemos crear un grupo de neuronas, A , que mira los pequeños segmentos de tiempo de nuestros datos.² A examina todos los segmentos, calculando ciertas características. Entonces, la salida de esta capa convolucional se alimenta a una capa completamente conectada, F

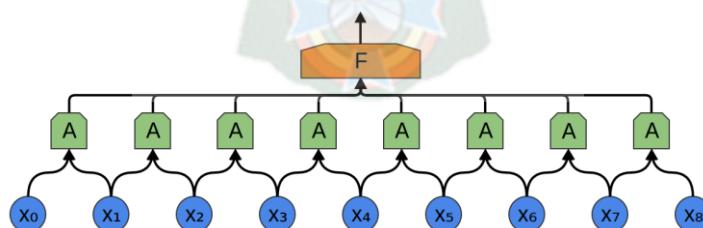


Figura 2.16: Ejemplo de una red neuronal convolucional 2D paso 3

Fuente: (Olah, 2014).

En el ejemplo anterior, A solo miraba segmentos que constaban de dos puntos. Esto no es realista. Por lo general, la ventana de una capa de convolución sería mucho más grande.

En el siguiente ejemplo, A mira 3 puntos. Eso tampoco es realista. Por desgracia, es difícil visualizar A conectarse a muchos puntos.

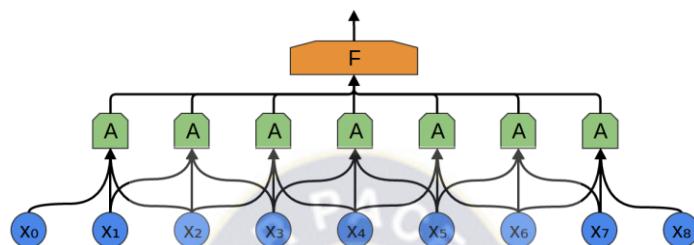


Figura 2.17: Ejemplo de una red neuronal convolucional 2D paso 4

Fuente: (Olah, 2014).

Una muy buena propiedad de las capas convolucionales es que son componibles. Puede alimentar la salida de una capa convolucional a otra. Con cada capa, la red puede detectar características de mayor nivel y más abstractas.

En el siguiente ejemplo, tenemos un nuevo grupo de neuronas, B. B se usa para crear otra capa convolucional apilada sobre la anterior.

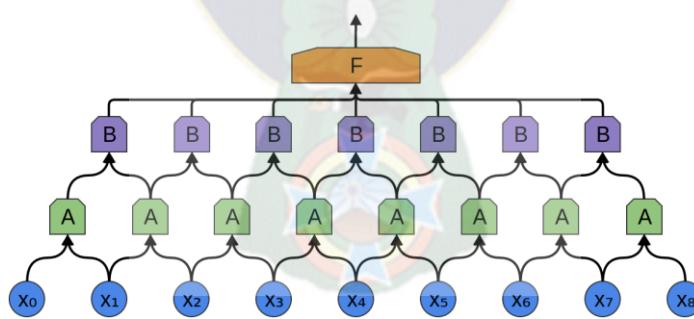


Figura 2.18: Ejemplo de una red neuronal convolucional 2D paso 5

Fuente: (Olah, 2014).

Las capas convolucionales a menudo se entrelazan con capas de agrupamiento. En particular, hay un tipo de capa llamada capa de acumulación máxima que es extremadamente popular.

A menudo, desde una perspectiva de alto nivel, no nos importa el momento preciso en que una característica está presente. Si un cambio en la frecuencia ocurre un poco antes o después, ¿Importa?

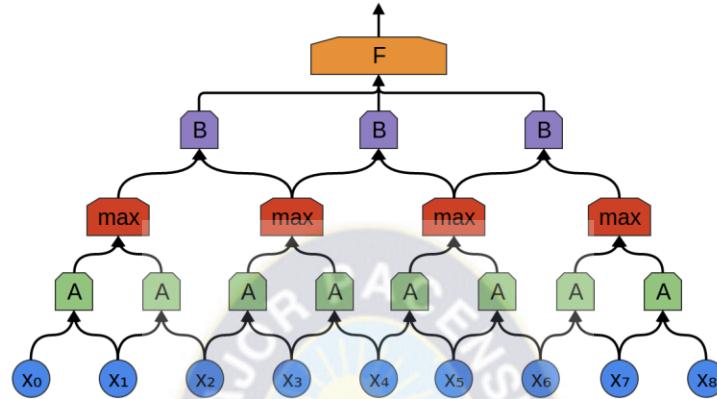


Figura 2.19: Ejemplo de una red neuronal convolucional 2D paso 6

Fuente: (Olah, 2014).

En nuestros ejemplos anteriores, hemos utilizado capas convolucionales de 1 dimensión. Sin embargo, las capas convolucionales también pueden funcionar en datos de mayor dimensión. De hecho, los éxitos más famosos de las redes neuronales convolucionales son la aplicación de redes neuronales convolucionales 2D para el reconocimiento de imágenes (Olah, 2014).

En una capa convolucional bidimensional, en lugar de mirar segmentos, A ahora verá parches.

Para cada parche, A calculará las características. Por ejemplo, podría aprender a detectar la presencia de un borde. O podría aprender a detectar una textura. O tal vez un contraste entre dos colores (Olah, 2014).

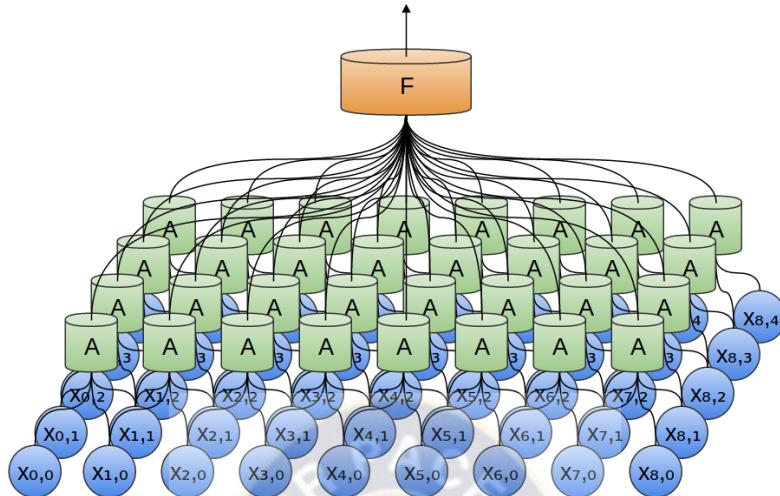


Figura 2.20: Ejemplo de una red neuronal convolucional 2D paso 7

Fuente: (Olah, 2014).

En el ejemplo anterior, alimentamos la salida de nuestra capa convolucional a una capa completamente conectada. Pero también podemos componer dos capas convolucionales, como lo hicimos en el caso unidimensional.

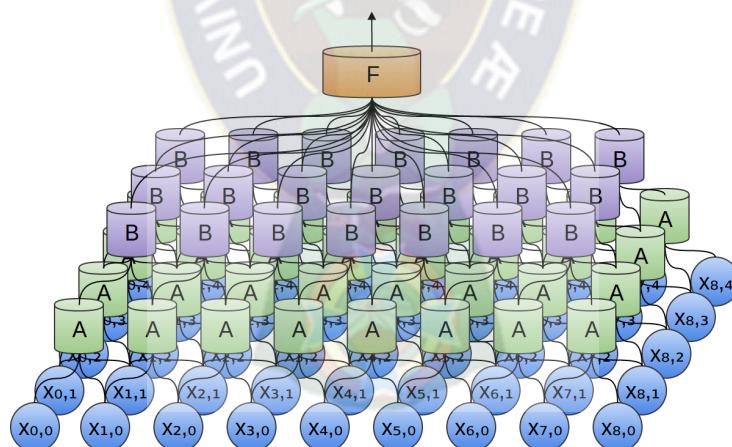


Figura 2.21: Ejemplo de una red neuronal convolucional 2D paso 8

Fuente: (Olah, 2014).

También podemos hacer la agrupación máxima en dos dimensiones. Aquí, tomamos el máximo de características sobre un pequeño parche.

Lo que realmente se reduce a esto es que, al considerar una imagen completa, no nos importa la posición exacta de un borde, hasta un píxel. Es suficiente para saber dónde está dentro de unos pocos píxeles.

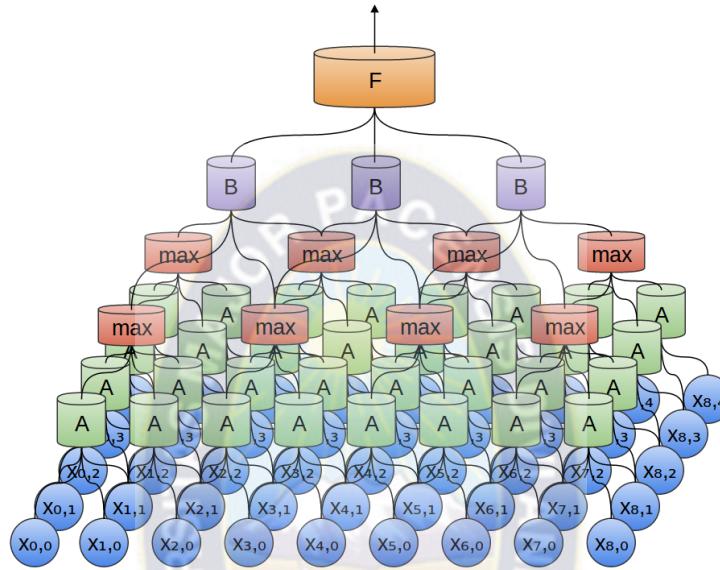


Figura 2.22: Ejemplo de una red neuronal convolucional 2D paso 9

Fuente: (Olah, 2014).

Las redes convolucionales tridimensionales también se usan a veces, para datos como videos o datos volumétricos (p. Ej., Escaneos médicos en 3D). Sin embargo, no son muy utilizados y mucho más difíciles de visualizar. Ahora, anteriormente dijimos que A era un grupo de neuronas. Deberíamos ser un poco más precisos acerca de esto: ¿qué es A exactamente?

En capas convolucionales tradicionales, A es un conjunto de neuronas en paralelo, que todas obtienen las mismas entradas y calculan diferentes características.

Por ejemplo, en una capa convolucional bidimensional, una neurona podría detectar bordes horizontales, otra podría detectar bordes verticales y otra podría detectar contrastes de color verde-rojo (Olah, 2014).

2.4 VISIÓN ARTIFICIAL

El sistema visual humano es uno de los mecanismos de procesamiento de imágenes más poderosos que existen. Este sistema es capaz de detectar, analizar y almacenar imágenes con un gran poder de procesamiento. La visión es sin duda el sentido más empleado por la especie humana, y por lo mismo frecuentemente se olvida de su importancia.

La visión artificial o visión por computador es la ciencia y la tecnología que permite a las "máquinas" ver, extraer información de las imágenes digitales, resolver alguna tarea o entender la escena que están visionando (Maduell, 2015).

La visión artificial es una disciplina científica que engloba todos los procesos y elementos que proporcionan ojos a una máquina y se podría decir que: "la visión artificial o comprensión de imágenes describe la deducción automática de la estructura y propiedades de un mundo tridimensional posiblemente dinámico, bien a partir de una o varias imágenes bidimensionales de ese mundo". Las estructuras y propiedades del mundo tridimensional que se quieren deducir en visión artificial incluyen no sólo sus propiedades geométricas, sino también sus propiedades materiales. Ejemplos de propiedades de los materiales son su color, iluminación, textura y composición. Si el mundo se modifica en el proceso de formación de la imagen, se necesitará inferir también la naturaleza del cambio, e incluso predecir el futuro. (EDMANS, 2006).

El hombre ha imitado muchas veces, en la construcción de sus artefactos, a la Naturaleza. En este caso también se cumple. Las cámaras de vídeo con sus ópticas hacen las veces del globo ocular, mientras el computador realizará las tareas de procesamiento, emulando el comportamiento del cerebro. Cuando se establecieron en la década de los 50, los objetivos de la Inteligencia Artificial, se suponía que con la llegada del siglo XXI habría máquinas que serían capaces de describir, con información de alto nivel, las escenas capturadas. Con el paso del tiempo se vio que aquel anhelo se iba desvaneciendo. Hoy en día, todavía no hay una teoría de la visión. No se conoce los mecanismos que el cerebro utiliza para obtener la información de la percepción. El cerebro es capaz, de manera inconsciente, de determinar la distancia a los objetos, de reconocerlos en diferentes posiciones, aunque se encuentren rotados y con información parcialmente oculta. En definitiva, el cerebro presenta una sofisticación en la percepción que ni ahora ni en mucho tiempo habrá posibilidad de implementar artificialmente.

Lo que si hace la Visión Artificial es construir nuevos y más sofisticados algoritmos que sean capaces de obtener información de bajo nivel visual. Y aunque todavía se esté años luz de la percepción visual de los seres vivos, la Visión Artificial es muy eficaz en tareas visuales repetitivas y alienantes para el hombre. Por ejemplo, en el campo de la inspección de productos en la industria o en contar células en una imagen de microscopía o en determinar la trayectoria de un vehículo en una autopista, etc. (Platero, 2009).

Resumiendo, las principales ventajas de la visión humana respecto a la artificial y viceversa, son (Platero, 2009):

Sistema humano:

- Mejor reconocimiento de objetos.
- Mejor adaptación a situaciones imprevistas.
- Utilización de conocimiento previo.
- Mejor en tareas de alto nivel de proceso.

Sistema artificial:

- Mejor midiendo magnitudes físicas.
- Mejor para la realización de tareas rutinarias.
- Mejor en tareas de bajo nivel de proceso.

2.4.1 ETAPAS UN PROCESO DE VISIÓN ARTIFICIAL

La visión artificial lleva asociada una enorme cantidad de conceptos relacionados con hardware, software y también con desarrollos teóricos. En esta sección se verán los pasos fundamentales, recogidos en la Figura 2.23, para llevar a cabo una tarea de visión artificial.

El primer paso en el proceso es adquirir la imagen digital. Para ello se necesitan sensores y la capacidad para digitalizar la señal producida por el sensor.

Una vez que la imagen digitalizada ha sido obtenida, el siguiente paso consiste en el preprocesamiento de dicha imagen. El objetivo del preprocesamiento es mejorar la imagen de forma que el objetivo final tenga mayores posibilidades de éxito.

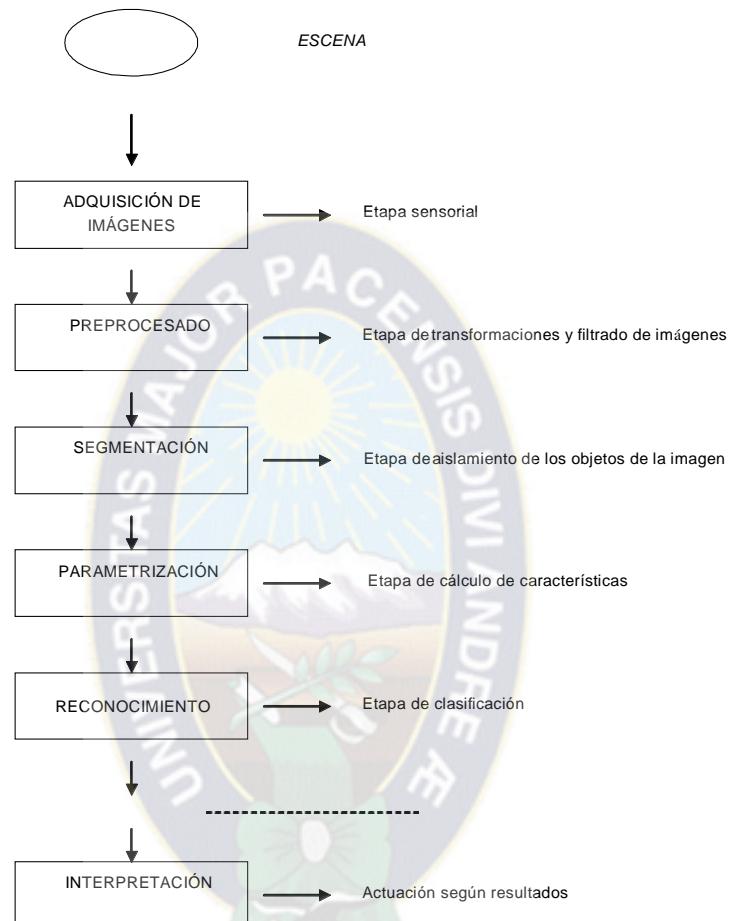


Figura 2.23: Diagrama de bloques de las etapas de un sistema de visión artificial

Fuente: (EDMANS, 2006).

El paso siguiente es la segmentación. Definida en sentido amplio, su objetivo es dividir la imagen en las partes que la constituyen o los objetos que la forman. En general la segmentación autónoma es uno de los problemas más difíciles en el procesamiento de la imagen. Por una parte, una buena segmentación facilitará mucho la solución del problema; por otra, la segmentación errónea conducirá al fallo (EDMANS, 2006).

La salida del proceso de segmentación es una imagen de datos que, o bien contienen la frontera de la región o los puntos de ella misma. Es necesario convertir estos datos a una forma que sea apropiada para el ordenador. La primera decisión es saber si se va a usar la representación por frontera o región completa. La representación por la frontera es apropiada cuando el objetivo se centra en las características de la forma externa como esquinas o concavidades y convexidades. La representación por regiones es apropiada cuando la atención se centra en propiedades internas como la textura o el esqueleto. Sin embargo, en muchas aplicaciones ambas representaciones coexisten (EDMANS, 2006).

La elección de una representación es sólo una parte de la transformación de los datos de entrada. Es necesario especificar un método que extraiga los datos de interés. La parametrización, que recibe también el nombre de selección de rasgos se dedica a extraer rasgos que producen alguna información cuantitativa de interés o rasgos que son básicos para diferenciar una clase de objetos de otra (EDMANS, 2006).

En último lugar se encuentran el reconocimiento y la interpretación. El reconocimiento es el proceso que asigna una etiqueta a un objeto basada en la información que proporcionan los descriptores (clasificación). La interpretación lleva a asignar significado al conjunto de objetos reconocidos (EDMANS, 2006).

2.4.2 ADQUISICIÓN DE IMÁGENES

Debido a que las computadoras, dispositivos móviles inteligentes solo pueden procesar imágenes digitales, es necesario convertir las mismas a esta forma. Un digitalizador de imágenes debe ser capaz de dividir la imagen en pequeñas regiones llamados elementos de imagen o píxeles (EDMANS, 2006).

La primera etapa, dentro de un proceso de visión computacional es la etapa de adquisición. En este primer paso, se trata de conseguir que la imagen sea lo más adecuada posible para que se pueda continuar con las siguientes etapas (EDMANS, 2006).

Una correcta adquisición de la imagen supone un paso muy importante para que el proceso de reconocimiento tenga éxito. Dentro de esta etapa existen múltiples factores que atañen directamente al proceso de captura de la imagen, formados fundamentalmente por: el sistema hardware de visión artificial (cámara, óptica, tarjeta de adquisición de imagen, ordenador y

software) y el entorno y posicionamiento de los elementos (la iluminación, el fondo, posición correcta de la cámara, ruido eléctrico-óptico externo, etc.) (EDMANS, 2006).

En el proceso de obtención de imágenes digitales se distinguen dos etapas. La primera, conocida como captura, utiliza un dispositivo, generalmente óptico, con el que obtiene información relativa a una escena. En la segunda etapa, que se conoce como digitalización, se transforma esta información, que es una señal con una o varias componentes continuas, en la imagen digital, que es una señal con todas sus componentes discretas (Vélez, Moreno, Sánchez, 2002).

2.4.2.1 MODELOS DE CAPTURA DE IMÁGENES

A grandes rasgos, para capturar una imagen se suele distinguir entre dispositivos pasivos (basados generalmente en el principio de cámara oscura) y dispositivos activos (basados en el escaneo), aunque también se pueden realizar mediciones de propiedades sobre un objeto y posteriormente construir una imagen de manera sintética. (Vélez, et al., 2002)

2.4.2.2 LA DIGITALIZACIÓN

Es el proceso de paso del mundo continuo (o analógico) al mundo discreto (o digital). En la digitalización normalmente se distinguen dos procesos: el *muestreo* (“sampling”) y la *cuantización* (“quantization”) (Vélez, et al., 2002).

2.4.2.3 REPRESENTACIÓN DE LA IMAGEN Y ESTRUCTURAS DE DATOS

Las imágenes suelen almacenarse en los ordenadores en forma de ficheros. En este punto se analizarán las estructuras que se usan a tal efecto, los métodos utilizados para optimizar el espacio requerido y algunos de los diferentes formatos estándar (TIFF, GIF, BMP, JPG...).

Estructura del fichero de imagen

Generalmente una imagen almacenada en un ordenador está constituida (ver Figura 2.24) por un mapa de bits (sería mejor decir de píxeles) precedido por una cabecera que describe sus características (tamaño de la imagen, modo de color, paleta, resolución de la imagen...). Frecuentemente, cuando la imagen se encuentra en la memoria principal del ordenador la cabecera y el mapa de bits no están contiguos (Vélez, et al., 2002).

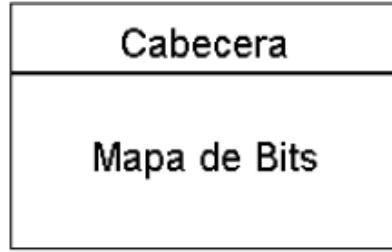


Figura 2.24: Esquema de un fichero gráfico.

Fuente: (Vélez, et al., 2002).

Compresión de imágenes

En ocasiones es impracticable el tratamiento directo de ciertas imágenes debido a la gran cantidad de datos que requiere su almacenamiento o su transmisión. La *compresión* de las imágenes trata este problema, mediante la reducción de la cantidad de datos necesarios para representar una imagen digital (Vélez, et al., 2002).

El primer punto que se debe tener en cuenta es que no son lo mismo “los datos usados para presentar una imagen” que “la información que contienen”. Por ejemplo: la idea del número “dos” se puede representar como: 2, dos, II, 6-3-1... Cada representación necesita diferente espacio para su codificación (1, 3, 2 y 5 caracteres respectivamente) pero codifican la misma información (Vélez, et al., 2002).

Si n_1 y n_2 denotan el tamaño de los datos necesarios para representar la misma información en dos sistemas diferentes s_1 y s_2 , definimos la *razón de compresión* C_R de n_1 frente a n_2 como:

$$C_R = \frac{n_1}{n_2}$$

Se define la redundancia relativa R_D como:

$$R_D = 1 - \frac{1}{C_R}$$

Estudiando estas fórmulas se observa que:

Si $n_1 = n_2 \Rightarrow C_R = 1 \Rightarrow R_D = 0$ (No hay redundancia de n_1 respecto a n_2)

Si $n_1 \gg n_2 \Rightarrow C_R \rightarrow \infty \Rightarrow R_D \rightarrow 1$ (n_1 es muy redundante respecto a n_2)

$$\text{Si } n_1 \ll n_2 \Rightarrow C_R \rightarrow 0 \Rightarrow R_D \rightarrow -\infty \quad (n_2 \text{ es muy redundante respecto a } n_1)$$

Así, se dice que un código es más redundante que otro cuando precisa más datos que aquél para describir la misma información. Clásicamente se distinguen tres tipos de redundancia:

- Redundancia en la codificación
- Redundancia en la representación espacial de los píxeles
- Redundancia visual.

Formatos comerciales de representación

Existen multitud de formatos de ficheros de imágenes de tipo mapa de bits. Se puede hablar de ficheros tipo BMP, TIFF, GIF, JFIF, PGM... Cada uno ofrece ciertas ventajas que otros formatos pueden no contemplar. La Tabla 2.1 recoge las principales características de algunos de estos formatos (Vélez, et al., 2002).

Formato	Color Real	Paleta	Grises	Bitonal	Compresión	Origen	Multi-Imagen
Bitmap	SI	SI	SI	SI	Run-Length	Windows	NO
TIFF	SI	SI	SI	SI	JPG, LZW, Runs, CCITT4, CCITT3, PackBits	Estándar	SI
JFIF	SI	NO	SI	NO	JPEG	Estándar	NO
PCX	NO	SI	NO	NO	Propia	Windows	NO
PGM	NO	NO	SI	NO	NO	Unix	NO
GIF	NO	SI	SI	SI	LZW	Estándar	SI

Tabla 2.1: Diferentes formatos para ficheros gráficos y características principales.

Fuente: (Vélez, et al., 2002).

2.4.3 PREPROCESADO

En este apartado se tratan las operaciones y transformaciones que se aplican sobre las imágenes digitales en una etapa de procesamiento previa a las de segmentación y al reconocimiento. Su

objetivo es mejorar o destacar algún elemento de las imágenes, de manera que las etapas posteriores sean posibles o se simplifiquen (Vélez, et al., 2002).

Todas las operaciones que se van a describir en esta etapa se pueden explicar desde la perspectiva ofrecida por la teoría de filtros. Un filtro puede verse como un mecanismo de cambio o transformación de una señal de entrada a la que se le aplica una función, conocida como función de transferencia, para obtener una señal de salida. En este contexto se entiende por señal una función de una o varias variables independientes. Los sonidos y las imágenes son ejemplos típicos de señales (Vélez, et al., 2002).



Figura 2.25: Esquema de funcionamiento de un filtro.

Fuente: (Vélez, et al., 2002).

En el diagrama anterior se representa el esquema general de funcionamiento de un filtro, siendo E la función de entrada, S la de salida y H la función de transferencia del filtro. Todas estas señales y funciones pueden ser discretas o continuas, y aunque en el tratamiento de imágenes se procesan señales y funciones discretas, suele recurrirse al caso continuo para explicar sus comportamientos, ya que sobre las funciones continuas es posible emplear herramientas más potentes de cálculo matemático (Vélez, et al., 2002).

En el resto del capítulo se describe las principales operaciones que se puede realizar sobre las imágenes digitales. La mayoría de las explicaciones se realizarán, por simplicidad sobre imágenes en niveles de gris. Su extensión a imágenes en color (RGB) suele consistir en repetir el tratamiento que se describe para cada una de las componentes de color. En los casos en que esto no sea posible se indicará el procedimiento adecuado.

2.4.3.1 FILTRADO ESPACIAL

Se puede demostrar matemáticamente que la transformación al dominio de la frecuencia (F), la aplicación del filtro (H) sobre los coeficientes, y la aplicación de la inversa de la transformada

(F^{-1}) , se puede realizar directamente en el dominio del espacio. Esta operación en el dominio del espacio se conoce como *convolución*, y exige el conocimiento de la transformada inversa (h) del filtro (Vélez, et al., 2002).

En el caso bidimensional discreto, la transformada inversa (h) del filtro resulta una matriz de tamaño $N \times N$, que es la función de transferencia que se conoce como *función impulsional*. Así, la operación de convolución queda:

$$I'(x, y) = I(x, y) * h(x, y) = \sum_{i=-\frac{N}{2}}^{\frac{N}{2}} \sum_{j=-\frac{N}{2}}^{\frac{N}{2}} I(i, j) \cdot h(x - i, y - j) \quad \forall x, y = 0, 1, \dots, N - 1$$

La carga computacional de esta operación es elevada. Sin embargo, es posible realizar aproximaciones a esta operación usando una función de transferencia h que tenga un número de elementos muy inferior a $N \times N$. Estas funciones impulsionales reducidas se denominan *funciones de filtrado espacial* (Vélez, et al., 2002).

Filtros paso bajo

El filtrado paso bajo espacial se basa en el promediado de los píxeles adyacentes al píxel que se evalúa. Quizás el filtro paso bajo más simple que se puede diseñar se corresponde con una matriz de 3×3 con todos los elementos a 1. El resultado se deberá dividir por 9 para obtener valores dentro del rango de la paleta. En la figura adjunta se puede apreciar el resultado de la aplicación de este filtro (Vélez, et al., 2002).

Las siguientes matrices de convolución definen otros filtros paso bajo:

$$h = \frac{1}{10} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad h = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Otro filtro paso bajo es el *filtro de la mediana*. Éste se basa en sustituir el valor de un píxel por el de la mediana del conjunto formado por el mismo y sus ocho vecinos.

El *filtro del bicho raro* es otro ejemplo de filtro paso bajo. Consiste en comparar la intensidad de un píxel con la de sus 8 vecinos. Si la diferencia es superior a cierto umbral U (que debe elegirse

previamente), se sustituye tal píxel por el valor promedio de los píxeles vecinos, en otro caso se mantiene su valor de intensidad (Vélez, et al., 2002).

Tanto el filtro de la mediana, como el filtro del “bicho raro” son filtros no lineales, es decir, no se pueden deducir de una convolución, y por tanto no tienen equivalente en el dominio de la frecuencia.

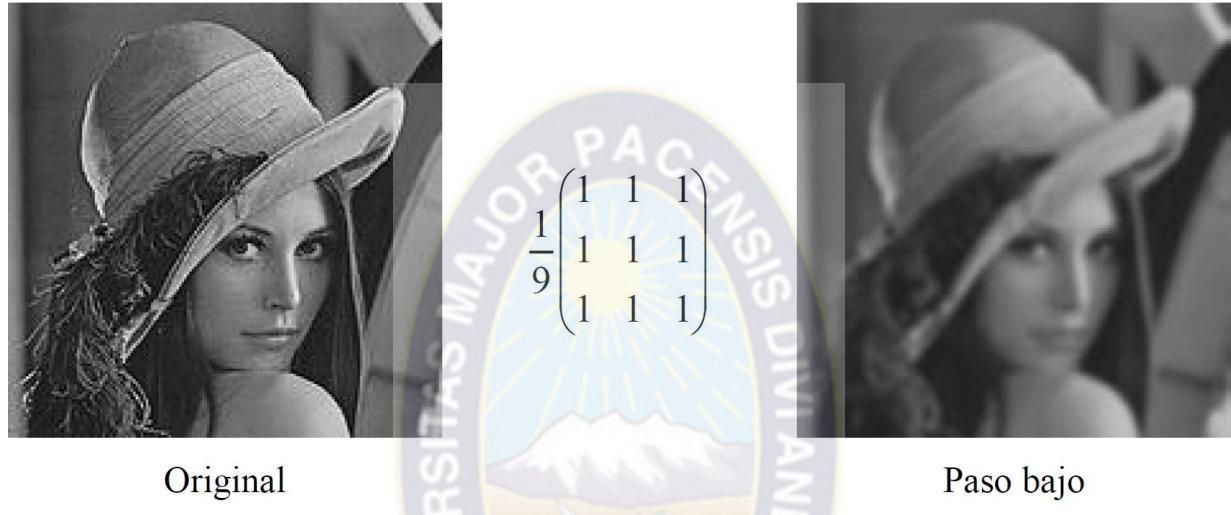


Figura 2.26: Aplicación de un filtrado espacial paso bajo

Fuente: (Vélez, et al., 2002).

Filtros paso alto

El cálculo de la derivada direccional de una función nos habla de cómo se producen los cambios en tal dirección. Tales cambios, que se corresponden con las altas frecuencias, suelen corresponder a los bordes de los objetos presentes en las imágenes (Vélez, et al., 2002).

Partiendo de que el operador gradiente se define como:

$$\nabla(I(x, y)) = \frac{\partial I}{\partial x} \vec{u}_x + \frac{\partial I}{\partial y} \vec{u}_y$$

Se definen los filtrados de convolución G_x , y G_y :

$$G_x = \frac{\partial I}{\partial x} = I(x, y) * h_1(x, y)$$

$$G_y = \frac{\partial I}{\partial y} = I(x, y) * h_2(x, y)$$

Obteniendo h_1 y h_2 mediante una aproximación a la derivada con la resta. Es decir, si se consideran los píxeles de la siguiente figura:

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Las derivadas serían:

$$\frac{\partial f}{\partial x} = z_5 - z_6 \quad \text{y} \quad \frac{\partial f}{\partial y} = z_5 - z_8$$

Obteniendo unas matrices de convolución h_1 y h_2 .

$$h_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{pmatrix} \quad h_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{pmatrix}$$

En el caso de que no se deseé considerar la dirección del vector gradiente, sino sólo su módulo, se escribe:

$$|I_G(x, y)| = \sqrt{G_x^2(x, y) + G_y^2(x, y)}$$

Con el fin de reducir la carga computacional la expresión anterior puede sustituirse por esta otra:

$$|I_G(x, y)| = \frac{1}{2} |G_x(x, y) + G_y(x, y)|$$

Una aproximación mejor al gradiente está dada por las expresiones:

$$\frac{\partial f}{\partial x} = (z_3 + 2z_6 - z_9) - (z_1 + 2z_4 - z_7)$$

$$\frac{\partial f}{\partial y} = (z_1 + 2z_2 - z_3) - (z_7 + 2z_8 - z_9)$$

Dando lugar a las matrices h_1 y h_2 .

$$h_1 = \frac{1}{4} \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \quad h_2 = \frac{1}{4} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

Estas matrices se conocen como *ventanas de Sobel*, que fue quien las propuso. Mediante ellas se calcula el gradiente en las direcciones horizontal y vertical. En la Figura 2.27 se ve cómo el resultado de aplicar h_1 sobre la imagen de Lena produce una imagen en la que aparecen los contornos horizontales de la figura de la imagen original (Vélez, et al., 2002).

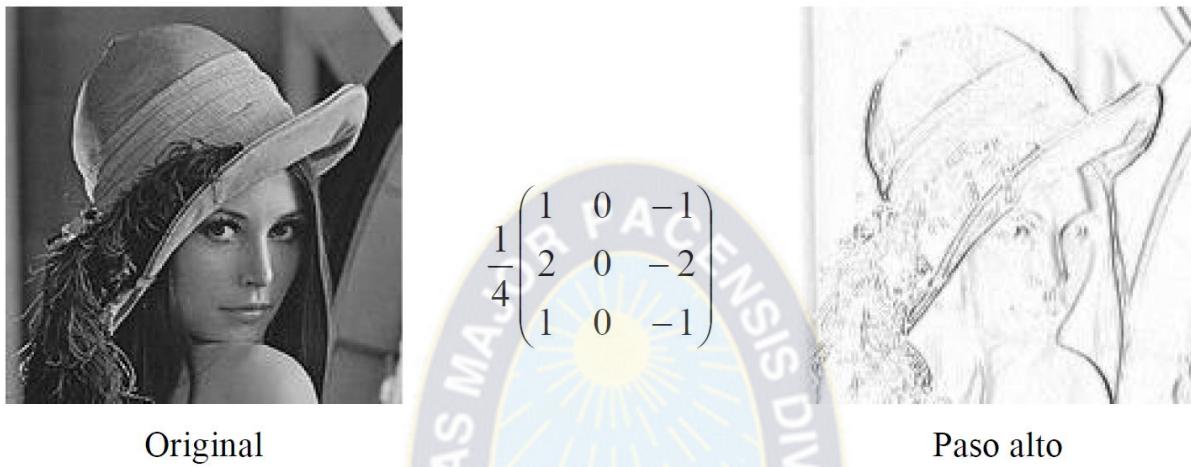


Figura 2.27: Filtrado paso alto de Sobel en la dirección x en valor absoluto. El resultado se presenta con el valor de los píxeles invertidos.

Fuente: (Vélez, et al., 2002).

2.4.4 SEGMENTACIÓN

La segmentación tiene como propósito realizar una partición de la imagen en regiones significativas. Los resultados serán utilizados en etapas sucesivas para su descripción, reconocimiento e interpretación (Vélez, et al., 2002).

Los métodos de segmentación asumen que las regiones a extraer poseen algunas características homogéneas distintivas.

El problema de la segmentación, por lo tanto, puede verse como un proceso de reconocimiento de patrones (regiones deseadas) o como un proceso de decisión, proceso cuyo fin último es establecer límites entre regiones.

El proceso de decisión utiliza en su forma más simple una única variable, por ejemplo, el nivel de gris de un pixel. Obviamente, para problemas más complejos se utilizan varias variables (Vélez, et al., 2002).

El punto de partida de la segmentación será la imagen pre tratada. A continuación, se dispone de dos alternativas

En caso que los objetos y el fondo tengan niveles de gris diferentes se procede directamente a la detección de umbrales con el fin de extraer cada uno de los objetos de la imagen. Esta técnica clasifica cada pixel como perteneciente al objeto o al fondo según corresponda su nivel de gris.

Sin embargo, en muchas ocasiones no resulta posible diferenciar de una forma tan sencilla los distintos elementos presentes en la imagen. En estas circunstancias se recurre al estudio de gradientes de luminosidad, habida cuenta de que los bordes de las distintas regiones, corresponden siempre a zonas de elevado contraste. Existen una extensa variedad de operadores que resaltan las zonas de alto gradiente de luminosidad, y que agrupan bajo el nombre genérico de “extractores de contorno”. Los puntos de la imagen resultante pueden ser fácilmente clasificados como pertenecientes o no a alguna frontera, mediante una bancarización (Vélez, et al., 2002).

2.4.5 PARAMETRIZACIÓN

A la salida del proceso de segmentación habitualmente se tienen los datos de pixeles en bruto, que constituyen bien el contorno de una región o bien todos los puntos de una región determinada.

En cada caso es necesario convertir los datos a una forma adecuada para el procesamiento por computadora. La primera decisión que hay que tomar es si los datos se han de representar como un contorno o como una región completa. La representación como un contorno es la adecuada cuando el interés en las características de la forma exterior, como esquinas e inflexiones. La representación regional es adecuada cuando el interés se centra en propiedades internas, como la textura o la estructura de un objeto. Sin embargo, en algunas aplicaciones ambas representaciones coexisten. Esto ocurre en las aplicaciones para reconocimiento de caracteres, que a menudo requieren algoritmos basados en la forma de los bordes, así como en la estructura y otras propiedades internas (Vélez, et al., 2002).

La elección de una representación es solo una parte de la solución para transformar los datos de pixeles en bruto a una forma adecuada para ser posteriormente tratados por computadora. También debe especificarse un método para describir los datos de forma que se resalten los rasgos de interés (Vélez, et al., 2002).

2.4.6 RECONOCIMIENTO E INTERPRETACIÓN

El reconocimiento es el proceso que asigna una etiqueta a un objeto basándose en la información proporcionada por sus descriptores. La interpretación implica asignar significado a un conjunto de objetos reconocidos (Vélez, et al., 2002).

El conocimiento sobre un dominio del problema está codificado en un sistema de procesamiento de imágenes como una base de datos de conocimiento. Este conocimiento puede ser tan simple como detallar las regiones de una imagen donde se sabe que se ubica información de interés, limitando si la búsqueda que ha de realizarse para hallar tal información (Vélez, et al., 2002).

En general, las funciones del procesamiento que incluyen reconocimiento e interpretación están asociadas con aplicaciones del análisis de imágenes en las que el objetivo es la extracción automática (o incluso parcialmente automática) de información de una imagen (Vélez, et al., 2002).

2.5 OPEN COMPUTER VISION LIBRARY

OpenCV (Open Source Computer Vision Library) es una biblioteca de software libre de visión por computador y software de aprendizaje automático. OpenCV fue construido para proporcionar una infraestructura común para aplicaciones de visión por computadora y para acelerar el uso de la percepción de la máquina en los productos comerciales. Al ser un producto con licencia de BSD, OpenCV facilita a las empresas utilizar y modificar el código (OpenCV team, 2018).

La biblioteca cuenta con más de 2500 algoritmos optimizados, que incluyen un conjunto completo de algoritmos clásicos y avanzados de visión por computador y aprendizaje automático. Estos algoritmos se pueden usar para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en videos, rastrear movimientos de la cámara, rastrear objetos en movimiento, extraer modelos 3D de objetos, producir nubes de puntos 3D desde cámaras estéreo, unir imágenes para producir una alta resolución imagen de una escena completa, encuentre imágenes similares de una base de datos de imágenes, elimine los ojos rojos de las imágenes tomadas con flash, siga los movimientos oculares, reconozca el escenario y establezca marcadores para superponerlo con realidad aumentada, etc. OpenCV tiene más de 47 mil personas de usuarios comunidad y número estimado de descargas que exceden los 14 millones. La biblioteca se usa ampliamente en compañías, grupos de investigación y por organismos gubernamentales (OpenCV team, 2018).

Junto con compañías bien establecidas como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda y Toyota que emplean la biblioteca, hay muchas nuevas empresas como Applied Minds, VideoSurf y Zeitera, que hacen un uso extensivo de OpenCV. Los usos implementados de OpenCV abarcan todo el rango, desde coser imágenes de streetview, detectar intrusiones en video de vigilancia en Israel, monitorear equipos mineros en China, ayudar a los robots a navegar y recoger objetos en Willow Garage, detectar accidentes de ahogamiento en piscinas en Europa, ejecutar arte interactivo en España y Nueva York, revisar las pistas de aterrizaje en busca de escombros en Turquía, inspeccionar etiquetas de productos en fábricas de todo el mundo para la detección rápida de rostros en Japón (OpenCV team, 2018).

Tiene interfaces C ++, Python, Java y MATLAB y es compatible con Windows, Linux, Android y Mac OS. OpenCV se inclina principalmente hacia las aplicaciones de visión en tiempo real y aprovecha las instrucciones de MMX y SSE cuando están disponibles. Un CUDA completo y OpenCL interfaces se están desarrollando activamente en este momento. Hay más de 500 algoritmos y aproximadamente 10 veces más funciones que componen o admiten esos algoritmos. OpenCV está escrito nativamente en C ++ y tiene una interfaz de plantilla que funciona a la perfección con contenedores STL (OpenCV team, 2018).

2.5.1 ESTRUCTURA Y CARACTERÍSTICAS DE LA LIBRERÍA OPENCV

La librería OpenCV está dirigida fundamentalmente a la visión por computador en tiempo real. Entre sus muchas áreas de aplicación destacarían: interacción hombre-máquina; segmentación y reconocimiento de objetos; reconocimiento de gestos; seguimiento del movimiento; estructura del movimiento; y robots móviles. También cuenta con una versión para programar en android.

2.5.2 HERRAMIENTAS DE LA LIBRERÍA OPENCV

La librería OpenCV proporciona varios paquetes de alto nivel para el desarrollo de aplicaciones de visión. Todos ellos se pueden agrupar en librerías de C/C++ dirigidas a usuarios avanzados y en herramientas descripting dirigidas, en este caso, a usuarios de nivel medio (ideal para practicar con las distintas técnicas de procesamiento de imágenes y visión).

2.5.3 OPENCV MANAGER

OpenCV Manager es un servicio Android orientado a gestionar la biblioteca OpenCV en los dispositivos de los usuarios finales. Permite el intercambio de librerías dinámicas OpenCV entre aplicaciones en el mismo dispositivo. Cualquier aplicación dependiente de OpenCV se instala desde Google Play o de forma manual;

Proporciona los siguientes beneficios:

- Menos uso de la memoria. Todas las aplicaciones utilizan los mismos binarios de servicio y no mantienen bibliotecas nativas dentro de sí mismos.
- optimizaciones específicas de hardware para todas las plataformas soportadas.
- Las Fuente de la biblioteca OpenCV son fiables. Todos los paquetes con OpenCV se publican en el mercado de Google Play.
- actualizaciones y correcciones de errores regular.

2.6 TENSORFLOW

A finales de noviembre de 2015 google liberó TensorFlow, una librería orientada a la construcción de modelos usando redes neuronales y usando todo el potencial del bicho que tengamos (Valiente, 2016).

Las cinco nociones básicas sobre TensorFlow que nos describe el propio Google:

- El diagrama de nuestro modelo está contenido dentro de la clase Graph.
- Se ejecuta en el contexto de Sessions.
- Los datos son tensores.
- Para mantener el estado durante las ejecuciones del diagrama, usaremos Variables.
- Para obtener los resultados y alimentar al sistema con variables externas, usaremos respectivamente fetches y feeds al correr la sesión.

TensorFlow es una biblioteca de software de código abierto para el cálculo numérico de alto rendimiento. Su arquitectura flexible permite una fácil implementación de computación en una

variedad de plataformas (CPU, GPU, TPU) y desde escritorios hasta clústeres de servidores y dispositivos móviles y periféricos. Desarrollado originalmente por investigadores e ingenieros del equipo Google Brain dentro de la organización AI de Google, cuenta con un sólido respaldo para el aprendizaje automático y el aprendizaje en profundidad, y el núcleo de computación numérica flexible se utiliza en muchos otros dominios científicos (Tensorflow, 2018).

Como menciona Sundar Pichai (CEO de Google, 2015). “TensorFlow es mucho más que Machine Learning, es la herramienta con la que podemos dar sentido a datos muy complejos. En Google utilizamos TensorFlow para todo, desde el reconocimiento de voz, hasta las respuestas inteligentes en Gmail o para buscar en Google Fotos. Nos permite construir y entrenar redes neuronales hasta cinco veces más rápido que con sistemas anteriores, por lo que podemos utilizarlo para mejorar nuestros productos mucho más rápidamente”.

2.6.1 FLUJO DE TRABAJO

Un grafo es un conjunto de operaciones animadas que representan nuestro modelo. Al iniciar cualquier operación, se añadirá al grafo(Graph) que está por defecto. Si no creamos ninguno, será el que TensorFlow no provee. Después, solo falta ejecutar el grafo a través de una sesión (Valiente, 2016).

2.6.2 TENSOR

Un Tensor es el bloque principal de trabajo al utilizar TensorFlow. Estos son como las variables que TensorFlow utiliza para trabajar con los datos. Cada tensor tiene dimensión y tipo.

La unidad central de datos en TensorFlow es el tensor. Un tensor consiste en un conjunto de valores primitivos formados en una matriz de cualquier cantidad de dimensiones. El rango de un tensor es su número de dimensiones, mientras que su forma es una tupla de enteros que especifica la longitud de la matriz a lo largo de cada dimensión. Aquí hay algunos ejemplos de valores de tensor (TensorFlow, 2018):

```
3. # a rank 0 tensor; a scalar with shape [],  
[1., 2., 3.] # a rank 1 tensor; a vector with shape [3]  
[[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]  
[[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]
```

2.6.3 GRAPH

Un gráfico computacional es una serie de operaciones TensorFlow dispuestas en un gráfico. El gráfico está compuesto por dos tipos de objetos.

Operations (u 'ops'): los nodos del gráfico. Las operaciones describen cálculos que consumen y producen tensores.

Tensors: los bordes en el gráfico. Estos representan los valores que fluirán a través del gráfico. La mayoría de las funciones de TensorFlow devuelven tf.Tensors.

Importante: tf.Los sensores no tienen valores, solo son identificadores de los elementos en el gráfico de computación.

Vamos a construir un gráfico computacional simple. La operación más básica es una constante. La función de Python que construye la operación toma un valor de tensor como entrada. La operación resultante no toma entradas. Cuando se ejecuta, muestra el valor que se pasó al constructor. Podemos crear dos constantes de punto flotante a y b de la siguiente manera (TensorFlow, 2018).:

```
a = tf.constant(3.0, dtype=tf.float32)
b = tf.constant(4.0) # also tf.float32 implicitly
total = a + b
print(a)
print(b)
print(total)
```

Las declaraciones de impresión producen:

```
Tensor("Const:0", shape=(), dtype=float32)
Tensor("Const_1:0", shape=(), dtype=float32)
Tensor("add:0", shape=(), dtype=float32)
```

Tenga en cuenta que la impresión de los tensores no genera los valores 3.0, 4.0 y 7.0 como cabría esperar. Las declaraciones anteriores solo construyen el gráfico de computación. Estos objetos tf.Tensor solo representan los resultados de las operaciones que se ejecutarán. (TensorFlow, 2018).

Cada operación en un gráfico recibe un nombre único. Este nombre es independiente de los nombres a los que están asignados los objetos en Python. Los tensores se nombran después de la operación que los produce, seguidos por un índice de salida, como en 'add:0' arriba (TensorFlow, 2018).

2.6.4 TENSORBOARD

TensorFlow proporciona una utilidad llamada TensorBoard. Una de las muchas capacidades de TensorBoard es visualizar un gráfico de computación. Puede hacerlo fácilmente con unos simples comandos (TensorFlow, 2018).

Primero guarda el gráfico de computación en un archivo de resumen TensorBoard de la siguiente manera:

```
writer = tf.summary.FileWriter('.')
writer.add_graph(tf.get_default_graph())
```

Esto producirá un archivo de evento en el directorio actual con un nombre en el siguiente formato:

```
events.out.tfevents.{timestamp}.{hostname}
```

Ahora, en una nueva terminal, inicie TensorBoard con el siguiente comando de shell:

```
tensorboard --logdir .
```

A continuación, abra la página de gráficos de TensorBoard en su navegador, y debería ver un gráfico similar al siguiente:



Figura 2.28: Ejemplo TensorBoard

Fuente: (TensorFlow, 2018).

2.6.5 SESSION

Para evaluar los tensores, crea una instancia de un objeto *tf.Session*, informalmente conocido como sesión. Una sesión encapsula el estado del tiempo de ejecución de TensorFlow y ejecuta las

operaciones de TensorFlow. Si un *tf.Graph* es como un archivo .py, una *tf.Session* es como el ejecutable de Python (TensorFlow, 2018).

El siguiente código crea un objeto *tf.Session* y luego invoca su método de ejecución para evaluar el tensor total que creamos anteriormente:

```
sess = tf.Session()
print(sess.run(total))
```

Cuando solicita la salida de un nodo con *Session.run* TensorFlow realiza una copia de seguridad a través del gráfico y ejecuta todos los nodos que proporcionan entrada al nodo de salida solicitado. Entonces esto imprime el valor esperado de 7.0:

```
print(sess.run({'ab':(a, b), 'total':total}))
```

que devuelve los resultados en una estructura del mismo diseño:

```
{'total': 7.0, 'ab': (3.0, 4.0)}
```

2.6.6 ENTRENAMIENTO

TensorFlow proporciona optimizadores que implementan algoritmos de optimización estándar. Estos se implementan como subclases de *tf.train.Optimizer*. Cambian incrementalmente cada variable para minimizar la pérdida. El algoritmo de optimización más simple es el descenso de gradiente, implementado por *tf.train.GradientDescentOptimizer*. Modifica cada variable de acuerdo con la magnitud de la derivada de pérdida con respecto a esa variable. Por ejemplo (TensorFlow, 2018):

```
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)
```

Este código crea todos los componentes del gráfico necesarios para la optimización y devuelve una operación de entrenamiento. Cuando se ejecuta, el entrenamiento actualizará las variables en el gráfico. Puede ejecutarlo de la siguiente manera:

```
for i in range(100):
    _, loss_value = sess.run((train, loss))
    print(loss_value)
```

Como el entrenamiento es una operación, no un tensor, no devuelve un valor cuando se ejecuta. Para ver la progresión de la pérdida durante el entrenamiento, ejecutamos el tensor de pérdida al mismo tiempo, produciendo resultados como los siguientes:

1.35659
1.00412
0.759167
0.588829
0.470264
0.387626
0.329918
0.289511
0.261112
0.241046
...

2.7 SISTEMA OPERATIVO ANDROID

Android es un sistema operativo móvil desarrollado por Google, es uno de los más conocidos junto con iOS de Apple. Está basado en Linux, que junto con aplicaciones middleware está enfocado para ser utilizado en dispositivos móviles como teléfonos inteligentes, tablets, Google TV y otros dispositivos (Basterra, Berte, Botello, Castillo & Venturi, 2016).

2.7.1 CARACTERÍSTICAS

Las características principales de Android son las siguientes: (Basterra, Berte, Botello, Castillo & Venturi, 2016).

- Código abierto.
- Núcleo basado en el Kernel de Linux.
- Adaptable a muchas pantallas y resoluciones.
- Utiliza SQLite para el almacenamiento de datos.
- Ofrece diferentes formas de mensajería.
- Navegador web basado en WebKit incluido.
- Soporte de Java y muchos formatos multimedia.
- Soporte de HTML, HTML5, Adobe Flash Player, etc.
- Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software.
- Catálogo de aplicaciones gratuitas o pagas en el que pueden ser descargadas e instaladas (Google Play).
- Bluetooth.
- Google Talk desde su versión HoneyComb, para realizar video llamadas.

- Multitarea real de aplicaciones.

2.7.2 ARQUITECTURA

Android es una pila de software de código abierto basado en Linux creada para una variedad amplia de dispositivos y factores de forma. En la siguiente figura 2.29 se muestran los componentes principales de la plataforma Android.

2.7.2.1 KERNEL DE LINUX

La base de la plataforma Android es el kernel de Linux. Por ejemplo, el tiempo de ejecución de Android (ART) se basa en el kernel de Linux para funcionalidades subyacentes, como la generación de subprocessos y la administración de memoria de bajo nivel.

El uso del kernel de Linux permite que Android aproveche funciones de seguridad claves y, al mismo tiempo, permite a los fabricantes de dispositivos desarrollar controladores de hardware para un kernel conocido (Android Developers, 2018).

2.7.2.2 CAPA DE ABSTRACCIÓN DE HARDWARE (HAL)

La capa de abstracción de hardware (HAL) brinda interfaces estándares que exponen las capacidades de hardware del dispositivo al framework de la Java API de nivel más alto. La HAL consiste en varios módulos de biblioteca y cada uno de estos implementa una interfaz para un tipo específico de componente de hardware, como el módulo de la cámara o de bluetooth. Cuando el framework de una API realiza una llamada para acceder a hardware del dispositivo, el sistema Android carga el módulo de biblioteca para el componente de hardware en cuestión (Android Developers, 2018).

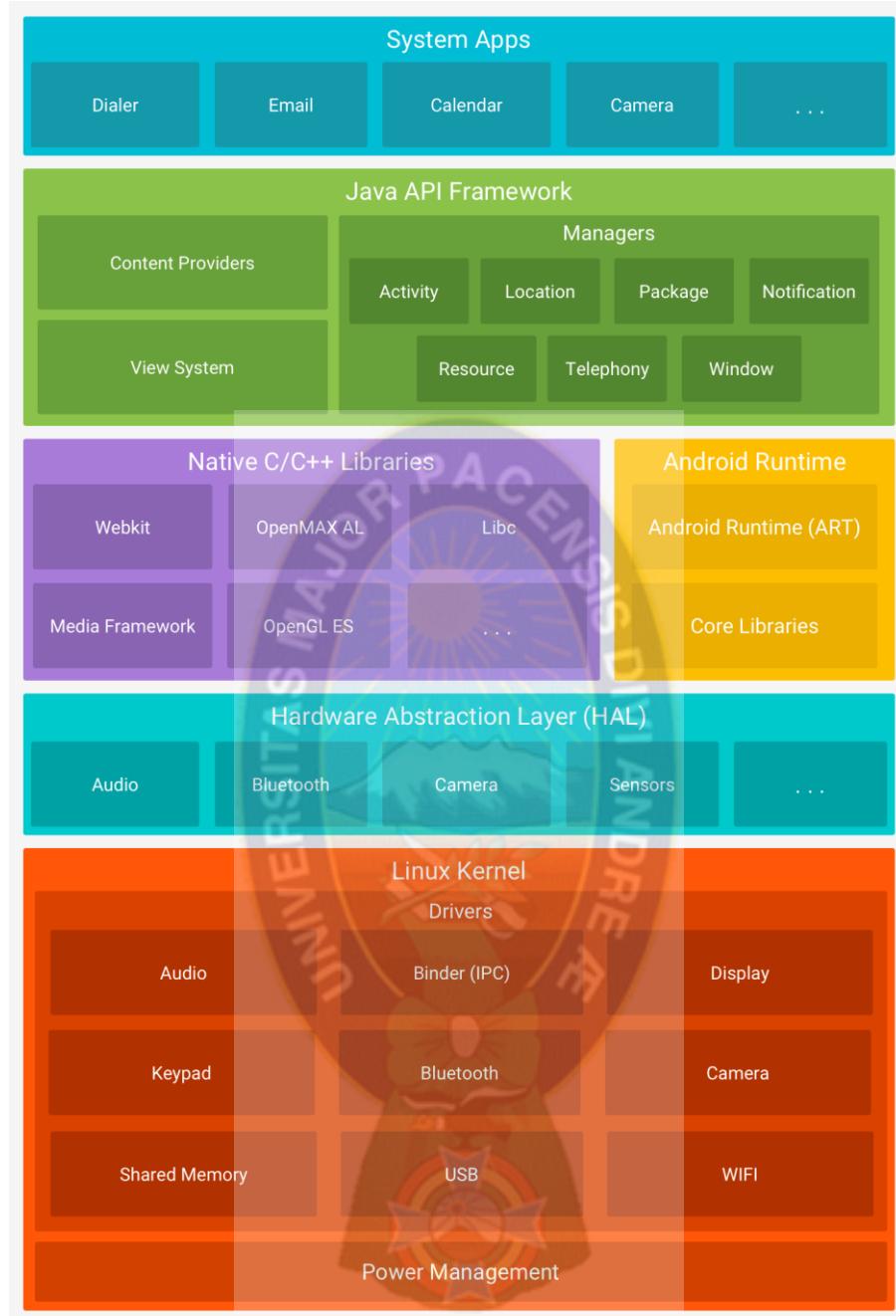


Figura 2.29: Arquitectura de Software de Android

Fuente: (Android Developers, 2018).

2.7.2.3 TIEMPO DE EJECUCIÓN DE ANDROID

Para los dispositivos con Android 5.0 (nivel de API 21) o versiones posteriores, cada app ejecuta sus propios procesos con sus propias instancias del tiempo de ejecución de Android (ART). El

ART está escrito para ejecutar varias máquinas virtuales en dispositivos de memoria baja ejecutando archivos DEX, un formato de código de bytes diseñado especialmente para Android y optimizado para ocupar un espacio de memoria mínimo. Crea cadenas de herramientas, como Jack, y compila fuentes de Java en código de bytes DEX que se pueden ejecutar en la plataforma Android.

Estas son algunas de las funciones principales del ART:

- compilación ahead-of-time (AOT) y just-in-time (JIT);
- recolección de elementos no usados (GC) optimizada;
- mejor compatibilidad con la depuración, como un generador de perfiles de muestras dedicado, excepciones de diagnóstico detalladas e informes de fallos, y la capacidad de establecer puntos de control para controlar campos específicos.

Antes de Android 5.0 (nivel de API 21), Dalvik era el tiempo de ejecución del sistema operativo. Si tu app se ejecuta bien en el ART, también debe funcionar en Dalvik, pero es posible que no suceda lo contrario.

En Android también se incluye un conjunto de bibliotecas de tiempo de ejecución centrales que proporcionan la mayor parte de la funcionalidad del lenguaje de programación Java; se incluyen algunas funciones del lenguaje Java 8, que el framework de la Java API usa (Android Developers, 2018).

2.7.2.4 BIBLIOTECAS C/C++ NATIVAS

Muchos componentes y servicios centrales del sistema Android, como el ART y la HAL, se basan en código nativo que requiere bibliotecas nativas escritas en C y C++. La plataforma Android proporciona la API del framework de Java para exponer la funcionalidad de algunas de estas bibliotecas nativas a las apps. Por ejemplo, puedes acceder a OpenGL ES a través de la Java OpenGL API del framework de Android para agregar a tu app compatibilidad con los dibujos y la manipulación de gráficos 2D y 3D.

Si desarrollas una app que requiere C o C++, puedes usar el NDK de Android para acceder a algunas de estas bibliotecas de plataformas nativas directamente desde tu código nativo (Android Developers, 2018).

2.7.2.5 FRAMEWORK JAVA API

Todo el conjunto de funciones del SO Android está disponible mediante API escritas en el lenguaje Java. Estas API son los cimientos que necesitas para crear apps de Android simplificando la reutilización de componentes del sistema y servicios centrales y modulares, como los siguientes:

- Un sistema de vista enriquecido y extensible que puedes usar para compilar la IU de una app; se incluyen listas, cuadrículas, cuadros de texto, botones e incluso un navegador web integrable.
- Un administrador de recursos que te brinda acceso a recursos sin código, como strings localizadas, gráficos y archivos de diseño.
- Un administrador de notificaciones que permite que todas las apps muestren alertas personalizadas en la barra de estado.
- Un administrador de actividad que administra el ciclo de vida de las apps y proporciona una pila de retroceso de navegación común.
- Proveedores de contenido que permiten que las apps accedan a datos desde otras apps, como la app de Contactos, o compartan sus propios datos.

Los desarrolladores tienen acceso total a las mismas API del framework que usan las apps del sistema Android (Android Developers, 2018).

2.7.2.6 APPS DEL SISTEMA

En Android se incluye un conjunto de apps centrales para correo electrónico, mensajería SMS, calendarios, navegación en Internet y contactos, entre otros elementos. Las apps incluidas en la plataforma no tienen un estado especial entre las apps que el usuario elige instalar; por ello, una app externa se puede convertir en el navegador web, el sistema de mensajería SMS o, incluso, el teclado predeterminado del usuario (existen algunas excepciones, como la app Settings del sistema).

Las apps del sistema funcionan como apps para los usuarios y brindan capacidades claves a las cuales los desarrolladores pueden acceder desde sus propias apps. Por ejemplo, si en tu app se intenta entregar un mensaje SMS, no es necesario que compiles esa funcionalidad tú mismo; como

alternativa, puedes invocar la app de SMS que ya está instalada para entregar un mensaje al receptor que especifiques (Android Developers, 2018).

2.7.3 API DE REDES NEURONALES

La API de Redes Neuronales de Android (NNAPI) es una API C de Android diseñada para ejecutar operaciones computacionalmente intensivas para el aprendizaje automático en dispositivos móviles. NNAPI está diseñado para proporcionar una capa básica de funcionalidad para los marcos de aprendizaje de máquina de alto nivel (como TensorFlow Lite, Caffe2 u otros) que construyen y entranen redes neuronales. La API está disponible en todos los dispositivos con Android 8.1 (API de nivel 27) o superior.

NNAPI admite la inferencia aplicando datos de dispositivos Android a modelos previamente entrenados y definidos por el desarrollador. Los ejemplos de inferencia incluyen la clasificación de imágenes, la predicción del comportamiento del usuario y la selección de las respuestas adecuadas a una consulta de búsqueda (Android Developers, 2018).

La inferencia en el dispositivo tiene muchos beneficios:

- Latencia: no necesita enviar una solicitud a través de una conexión de red y esperar una respuesta. Esto puede ser crítico para las aplicaciones de video que procesan tramas sucesivas provenientes de una cámara.
- Disponibilidad: la aplicación se ejecuta incluso cuando está fuera de la cobertura de la red.
- Velocidad: el nuevo hardware específico para el procesamiento de redes neuronales proporciona un cálculo significativamente más rápido que con la CPU de uso general sola.
- Privacidad: los datos no salen del dispositivo.
- Costo: no se necesita una granja de servidores cuando todos los cálculos se realizan en el dispositivo.

También hay compensaciones que un desarrollador debe tener en cuenta:

- Utilización del sistema: la evaluación de redes neuronales implica una gran cantidad de cálculos, lo que podría aumentar el uso de la batería. Debería considerar monitorear el

estado de la batería si esto es una preocupación para su aplicación, especialmente para cálculos de larga duración.

- Tamaño de la aplicación: preste atención al tamaño de sus modelos. Los modelos pueden ocupar varios megabytes de espacio. Si agrupar modelos grandes en su APK impactaría indebidamente a sus usuarios, es posible que desee considerar la posibilidad de descargar los modelos después de la instalación de la aplicación, el uso de modelos más pequeños o la ejecución de sus cálculos en la nube. NNAPI no proporciona funcionalidad para ejecutar modelos en la nube.

2.8 LENGUA DE SEÑAS

Es un sistema lingüístico cuyo medio es más visual que auditivo. Tiene su propio vocabulario, expresiones idiomáticas, gramática y sintaxis. Los parámetros formaciones de la lengua de señas son: Forma de la mano, lugar, movimiento, dirección del movimiento y expresión facial para ayudar a transmitir el significado del mensaje, siendo en esencia, una lengua viso gestual (Ministerio de Educación, 2016).

La Lengua de Señas Boliviana es propia de la comunidad Sorda Boliviana y está reconocida por el Estado Plurinacional de Bolivia mediante el Decreto Supremo N° 0328 del 14 de octubre del 2009 (Ministerio de Educación, 2016).

La lengua de señas es el camino para comprender y entender la identidad propia de la persona Sorda, en su mayoría los niños con sordera tienen dificultades para entender su identidad. Desde que nacen conviven con una familia oyente, pero deben necesariamente asistir a una unidad educativa especial y relacionarse con sus iguales para adquirir identidad y lengua, pues estos, si no se consolidan pueden influir en su desarrollo intelectual y social más adelante (Ministerio de Educación, 2016).

2.8.1 ALFABETO DACTIOLÓGICO

Es parte de la lengua de señas, aunque en sí mismo no son señas sino una representación manual para las diferentes letras de la lengua oral. El alfabeto sirve para comunicarse cuando no se conoce o no existe una seña. (Comité Nacional Contra el Racismo y toda forma de Discriminación, 2014).

En la figura 2.30 se muestra a detalle el Alfabeto Dactilológico de la Lengua de Señas Boliviana.

Alfabeto dactilológico



Figura 2.30 Alfabeto Dactilológico de la Lengua de Señas Boliviana

Fuente: (Ministerio de Educación, 2010).

2.8.2 DACTILOLOGÍA

La dactilología (alfabeto manual) para Sordos es lo primero que atrae a los oyentes para el aprendizaje de las lenguas de señas (Ministerio de Educación, 2010). Es la representación manual de cada una de las letras que componen el alfabeto. A través de ella se puede transmitir a la persona sorda cualquier que se desee comunicar, por complicada que ésta sea (Vilches, 2005).

El deletreo es una parte importante del sistema de comunicación de las personas sordas. Se trata, sencillamente, de la escritura del alfabeto castellano ejecutada en el aire en lugar de un papel. Existen veintinueve posiciones con sus variantes de movimiento de mano, algunas de las cuales son la representación exacta de la letra. El deletreo manual es usado en combinación con el lenguaje de signos para sustantivos, nombres propios, direcciones y palabras para las cuales no existe un ideograma o signo creado o es poco conocido por la comunidad signante, como ocurre con signos de reciente creación (nologismos) o palabras poco usuales. Su importancia no puede ser subestimada; es por tanto esencial para la persona que se inicie en la lengua de signos, concentrarse en desarrollar tanto las habilidades receptivas como las expresivas, con el fin de adquirir experiencia (Vilches, 2005).

Para realizar la dactilología, se utiliza la mano dominante (derecha para los diestros, e izquierda para los zurdos). Se ejecuta principalmente a la altura de la barbilla. Su realización se complementa con la articulación oral, por lo que es necesario que la cara y la boca sean visibles (Vilches, 2005).



Figura 2.31: Representación manual de la letra “A”

Fuente: (Sobre todo personas, 2013).

CAPÍTULO III

DISEÑO METODOLÓGICO

3.1 INTRODUCCIÓN

En el presente capítulo se presenta el diseño y la construcción de un sistema de reconocimiento e interpretación del alfabeto dactilológico para dispositivos móviles, es necesario seguir una metodología de construcción, en el presente trabajo se utiliza la metodología de desarrollo de sistemas de Visión Artificial definida en el capítulo 2.3 y utilizando métodos, librerías, algoritmos y técnicas que dispone la librería openCV para Android e implementaremos en la etapa de reconocimiento de Visión Artificial un modelo de Red Neuronal Artificial definida en el capítulo 2.2 para el reconocimiento de patrones de una señal del alfabeto dactilológico de la lengua de señas.

Previamente se realizará un análisis y explicación correspondiente de todo el material requerido para el desarrollo del presente trabajo de grado. Se describirá tanto el equipo requerido y necesario para la instalación de programas y la simulación de nuestra aplicación, como el software utilizado para lo cual realizaremos la instalación de las librerías de openCV en un entorno de trabajo de Android Studio.

En la figura 3.1 se detalla las etapas de desarrollo del sistema de visión artificial para el Reconocimiento e Interpretación del alfabeto dactilológico.

La primera etapa en el proceso es adquirir la imagen digital de una determinada señal del alfabeto dactilológico emitida por una persona a una distancia adecuada de la cámara. Para ello utilizaremos la cámara del dispositivo móvil con sistema operativo Android que realizará el proceso de adquisición de la imagen y almacenamiento en el dispositivo móvil para su posterior procesamiento.

Una vez que la imagen digitalizada ha sido obtenida, el siguiente paso consiste en el preprocesamiento de dicha imagen. El objetivo del preprocesamiento es mejorar la imagen de forma que el objetivo final tenga mayores posibilidades de éxito.



Figura 3.1: Etapas del sistema de Visión Artificial para el Reconocimiento e Interpretación del alfabeto dactilológico.

Fuente: Elaboración propia.

El paso siguiente es la segmentación. Definida en sentido amplio, su objetivo es dividir la imagen en las partes que la constituyen o los objetos que la forman. La salida del proceso de segmentación es una imagen de datos que, o bien contienen la frontera de la región o los puntos de ella misma.

Es necesario convertir estos datos a una forma que sea apropiada para el ordenador. La primera decisión es saber si se va a usar la representación por frontera o región completa. La representación por la frontera es apropiada cuando el objetivo se centra en las características de la forma externa como esquinas o concavidades y convexidades. La representación por regiones es apropiada cuando la atención se centra en propiedades internas como la textura o el esqueleto. Sin embargo, en muchas aplicaciones ambas representaciones coexisten.

La elección de una representación es sólo una parte de la transformación de los datos de entrada. Es necesario especificar un método que extraiga los datos de interés. La parametrización, que recibe también el nombre de selección de rasgos se dedica a extraer rasgos que producen alguna información cuantitativa de interés o rasgos que son básicos para diferenciar una clase de objetos de otra.

En último lugar se encuentran el reconocimiento y la interpretación. El reconocimiento es el proceso que asigna una etiqueta a un objeto basada en la información que proporcionan los descriptores (clasificación) para la cual se implementa y entrena un modelo de red neuronal convolucional para luego ser evaluada con datos de prueba. La interpretación lleva a asignar significado al conjunto de objetos reconocidos, en la cual se utiliza el altavoz del dispositivo móvil para la reproducción en voz sintética de la correspondiente señal.

3.2 MATERIAL REQUERIDO

En este apartado se realizará un análisis y explicación correspondiente de todo el material requerido para el desarrollo del presente trabajo de grado.

3.2.1 EQUIPO UTILIZADO

Para el presente trabajo de grado se utilizará un ordenador con sistema operativo Windows 10 de 64 bits con los siguientes los siguientes requisitos mínimos que exige Android Studio para el uso eficiente de este software, que viene siendo (Android Developers, 2018):

- Microsoft® Windows® 8/7/10 (32 o 64 bits).
- 3 GB de RAM como mínimo, 8 GB de RAM recomendado; más 1 GB para el emulador de Android.
- 2 GB de espacio disponible en disco mínimo, 4 GB recomendado (500 MB para IDE + 1,5 GB para Android SDK y la imagen del sistema emulador).
- 1280 x 800 resolución mínima de pantalla.
- Tener instalado Java Development Kit (JDK).

3.2.2 SOFTWARE REQUERIDO

A continuación, se detalla el software y versiones requeridas para el desarrollo del presente trabajo de grado.

- Python 3.6.5
- Tensorflow 1.2.1
- Matplotlib 2.0.2

- Numpy 1.13.0
- opencv-python 3.2.0.8
- Android Studio 3.1.2

3.3 ADQUISICIÓN DE LA IMAGEN

Este proceso es muy importante para reducir ruidos y otros factores que posiblemente afectan al sistema en los siguientes pasos de procesamiento de la imagen.

Como primer paso, se intenta que la imagen sea lo más adecuada posible para que se pueda continuar con las siguientes etapas.

Debido a esto y otras razones, por las que se optó por realizar las pruebas en un lugar donde las condiciones de luminosidad eran las adecuadas, y el medio también era el adecuado como para hacer la captura de las imágenes seguidamente procesar.

3.3.1 DISPOSITIVO DE CAPTURA

En este apartado vamos a identificar las características técnicas más relevantes del dispositivo móvil sobre el que nos apoyaremos para realizar las diferentes etapas del presente trabajo de grado y posteriormente identificaremos cuales de estas características deberían ser incorporadas o usadas por nuestra aplicación para que las imágenes capturadas sean obtenidas en las mejores condiciones posibles, y facilitar de este modo el proceso posterior a la adquisición, que es el procesamiento digital de la imagen.

El dispositivo móvil que utilizaremos a lo largo de todas las etapas del presente trabajo de grado es un dispositivo móvil de la marca Samsung modelo, *Samsung Galaxy J5 (2016)*.

A continuación, se identifican las características técnicas más relevantes que posee este dispositivo móvil (Samsung, 2016):

- *Procesador*
 - *Velocidad CPU* 1.2GHz
 - *Tipo CPU* Quad-Core
- *Pantalla*
 - *Tamaño* 5.0" (126.3mm)

- *Resolución* 720 x 1280 (HD)
 - *Tecnología* Super AMOLED
 - *Número de colores* 16M
 - *S Pen* No
- *Cámara*
 - *Cámara principal - Resolución* CMOS 13.0 MP (4128 x 3096 pixeles)
 - *Cámara principal - Autofocus* Sí
 - *Cámara frontal - Resolución* CMOS 5.0 MP
 - *Cámara principal - Flash* Sí
 - *Resolución de grabación de vídeo* FHD (1920 x 1080)@30fps
- *Memoria*
 - *RAM (GB)* 1.5 GB
 - *Memoria Interna (GB)* 8 GB
 - *Memoria Disponible* (GB)* 4.3 GB
 - *Externa MicroSD* (hasta 128GB)
- *Sistema Operativo* Android
- *Especificaciones físicas*
 - *Dimensiones (AlxAnxProf, mm)* 142.1 x 71.8 x 7.9
 - *Peso (g)* 146
- *Batería*
 - *Tiempo de uso de Internet (3G) (Horas)* Hasta 9
 - *Tiempo de uso de Internet (4G) (Horas)* Hasta 9
 - *Tiempo de uso de Internet (Wi-Fi) (Horas)* Hasta 11
 - *Tiempo de reproducción de Vídeo (Horas)* Hasta 13
 - *Capacidad (mAh)* 2600
 - *Extraible* Sí
 - *Tiempo de reproducción de Audio (Horas)* Hasta 62
 - *Tiempo en conversación (3G WCDMA) (Horas)* Hasta 18
- *Audio y Vídeo*
 - *Formatos de reproducción de Vídeo* MP4, M4V, 3GP, 3G2, MKV, WEBM

- *Resolución de reproducción de Vídeo FHD (1920 x 1080) @30fps*
- *Formatos de reproducción de Audio MP3, M4A, 3GA, AAC, OGG, OGA, WAV, AMR, AWB, FLAC, MID, MIDI, XMF, MXMF, IMY, RTTTL, RTX, OTA*

A continuación, se describen las características del procesador, la memoria RAM y la cámara principal del dispositivo móvil. Con la cámara de este dispositivo móvil van a ser tomadas las imágenes sobre las que posteriormente se realizará el reconocimiento e interpretación del alfabeto dactilológico de la lengua de señas. Hay que tener en cuenta que este dispositivo móvil posee también una cámara frontal de 5MP, aunque nosotros nos vamos a centrar en su cámara principal, ya que es la que se utilizará para adquirir las imágenes. Las características más importantes del procesador, la memoria RAM y la cámara principal son:

- Procesador
 - Velocidad CPU 1.2GHz
 - Tipo CPU Quad-Core
- Memoria RAM
 - RAM (GB) 1.5 GB
- Cámara Principal
 - 13.0 MP (4128 x 3096 pixeles)
 - Autofocus
 - Flash LED

3.3.2 DESARROLLO DE LA INTERFAZ DEL SISTEMA

3.3.2.1 USO DE PERMISOS

Una aplicación básica de Android no tiene permisos asociados de manera predeterminada. Esto significa que no puede hacer nada que afecte negativamente la experiencia del usuario o los datos en el dispositivo. Para usar funciones protegidas del dispositivo, debes incluir una o más etiquetas *<uses-permission>* en el manifiesto del app.

Ya que el sistema tiene que interactuar con la cámara del dispositivo se especifica lo siguiente en el manifiesto de la app.

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

3.3.2.2 INTERACCIÓN CON LA CAMARA DEL DISPOSITIVO

Otorgados los permisos correspondientes se procede al desarrollo de la interfaz de interacción con la cámara del dispositivo para realizar la adquisición de la imagen que será posteriormente procesada por el sistema. Para lo cual se desarrollaron las siguientes interfaces:

- Activity_camera.xml
- Camera_connection_fragment_stylize.xml
- Camera_connection_fragment_tracking.xml

Con estas interfaces el usuario interactúa con la aplicación móvil de reconociendo e interpretación del alfabeto dactilológico, como se puede apreciar en la figura 3.2 la interfaz se divide en dos partes, la parte superior y la parte inferior. En la parte superior en primera instancia se desplegará una lista ordenada de mayor a menor probabilidad de confianza del reconocimiento de la señal percibida por la cámara del dispositivo móvil, esta lista con probabilidades de acierto servirá para la etapa de análisis de resultados.

En la parte inferior se visualiza la imagen percibida por la cámara del dispositivo móvil junto a una delimitación de área para el mejor enfoque de una señal.

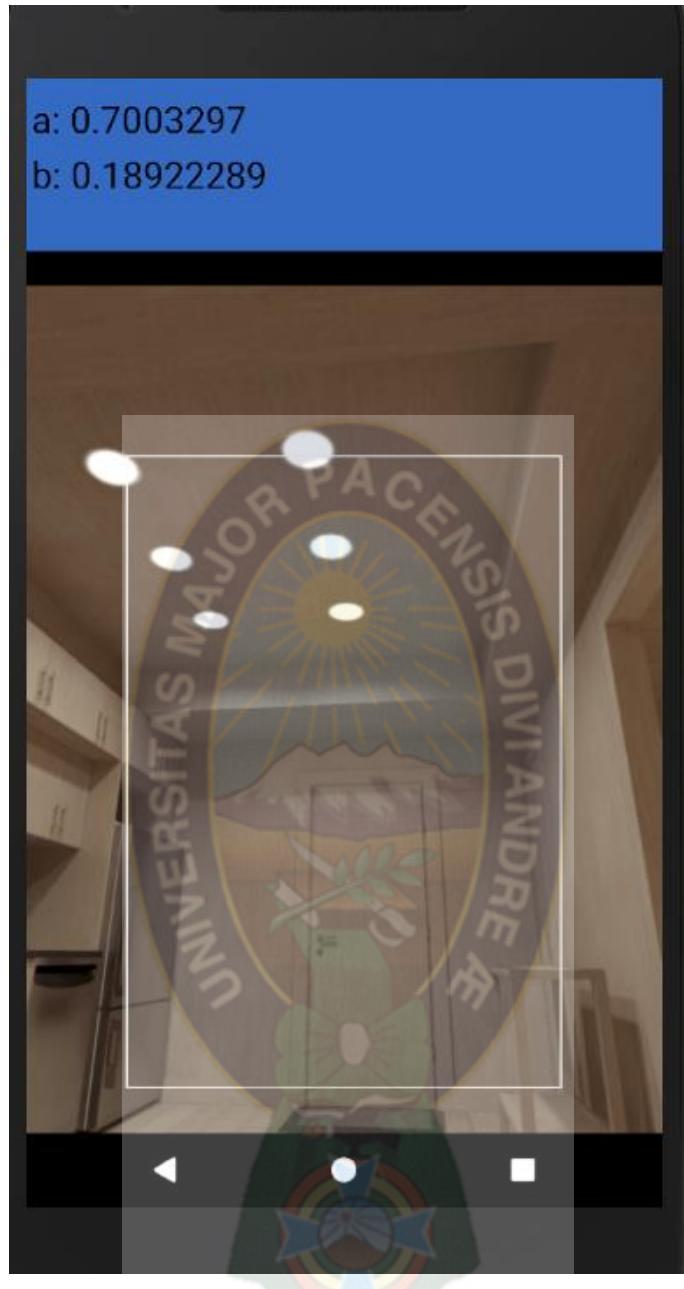


Figura 3.2: Interfaz de la aplicación móvil

Fuente: Elaboración propia.

3.4 PREPROCESADO

Una vez hecha la Adquisición de la imagen, continuamos con el preprocesado de la imagen, primero identificando que tipo de objetos queremos en nuestro proceso final llegar a ver, u obtener mediante los algoritmos de Visión Artificial.

Las técnicas de pre-procesado pretenden mejorar o realzar las propiedades de la imagen para facilitar las operaciones en las siguientes etapas.

Estas técnicas de pre-procesado se pueden dividir, en general, en varias facetas, para el prototipo se utilizó las siguientes:

- Realce o aumento del contraste (enhancement)
- Suavizado o eliminación del ruido (denoising)

La etapa de pre-procesamiento del sistema consiste en la manipulación de las imágenes que se desean traducir para obtener una representación adecuada de la señal para la etapa de procesamiento. La finalidad de esta etapa es eliminar toda aquella información innecesaria para el proceso de traducción y brindarle a la etapa de procesamiento el conjunto de datos de interés para la clasificación de cada una de las imágenes.

Esta etapa inicia realizando el procesamiento de la imagen, este procesamiento es realizado mediante los siguientes pasos: extracción de áreas de interés, eliminación de ruido y Recorte.

3.4.1 EXTRACCIÓN DE ÁREAS DE INTERÉS

La extracción de áreas de interés, reconoce el área conformada por la imagen, esta es la encargada de extraer todas aquellas áreas de la imagen donde podrá estar presente la mano. Para extraer las áreas de interés se emplea el algoritmo basado en la técnica Croma, con la que buscamos un área que contenga la señal en la imagen obtenida.

3.4.2 ELIMINACIÓN DE RUIDO

En algunos casos, al realizar la plantilla se genera ruido en la imagen, su eliminación es la encargada de continuar con el preprocesamiento. El resultado es una imagen binaria donde se obtiene la captura de la señal deseada.

Para lograr la eliminación de ruido aplicamos una convolución basada en filtros de Gauss a la imagen anterior, resultando una imagen estilizada con bordes difuminados. La finalidad de aplicar este filtro es hacer que los píxeles aislados de ruido presentes en la plantilla tomen valores considerablemente inferiores al blanco.

A partir de esta nueva imagen obtenida se procede a eliminar todas las áreas blancas excepto la de mayor tamaño. Y luego solo queda el área correspondiente a la señal de la mano.

3.5 SEGMENTACIÓN

Este proceso tiene como objetivo recortar la plantilla obtenida en la etapa anterior y a partir de ésta extraer de la imagen original la señal que realiza la mano con su información interna, a diferencia de la plantilla que representa únicamente el contorno de la forma de la mano.

Para realizar el corte del área, de la mano, analizamos cada pixel de la plantilla anteriormente creada.

El análisis consiste en la búsqueda de pixeles que contengan el blanco como valor, al encontrar esta coincidencia se toma las posiciones del pixel para copiar el valor del pixel de la imagen original y generar otra imagen que contiene solo la imagen de la señal. Luego la imagen obtenida es transformada a escala de grises para facilitar su manipulación, ya que al convertir la imagen RGB a escala de grises su representación pasa a ser de una matriz numérica de tres dimensiones ($N \times M \times 3$) a una matriz de una dimensión ($N \times M$).

A continuación, las dos imágenes recortadas son re escaladas a un tamaño preestablecido (100x100 px) que facilita el uso de la imagen en la etapa siguiente. El siguiente paso es generar una imagen con el guante en fondo negro. Esta imagen se logra haciendo una superposición de imágenes mediante la multiplicación entre la imagen binaria y la imagen en escala de grises obtenidas en la etapa de adquisición de la imagen. Finalmente se obtiene la imagen de la señal solo con las líneas del contorno para luego ajustar el tamaño de la imagen.

3.6 PARAMETRIZACIÓN

La etapa de procesamiento tiene como finalidad emplear las imágenes pre-procesadas para que éstos sean evaluados a través de un método de clasificación que determine si pertenece a alguna señal.

3.6.1 EXTRACCIÓN DE CARACTERÍSTICAS IMPORTANTES O RELEVANTES DE LA IMAGEN.

Tomamos la imagen obtenida al finalizar la etapa anterior y generamos una matriz de $n \times m$ que es representado como un vector.

Luego se encuentra el porcentaje de cada caja (100 pixeles por 100 pixeles) de la imagen que está vacío ya que negro tiene un valor de cero por lo tanto no sumará al área ocupada. Asignamos el porcentaje a una posición del vector que representa la letra. La matriz obtenida contiene las características más relevantes de la imagen.

3.7 RECONOCIMIENTO

Para realizar esta etapa, previamente se debe generar y entrenar a la red neuronal, esta etapa fue detallada en el capítulo 2.3. La red neuronal requerida se implementará haciendo uso del lenguaje de programación Python y TensorFlow.

Se utilizó la técnica de transferencia de aprendizaje, lo que significa que se comienza con un modelo que ya ha sido entrenado en otro problema. Luego lo reentrenamos en un problema similar. El aprendizaje profundo desde cero puede llevar días, pero el aprendizaje de transferencia se puede realizar en un menor tiempo.

Se utilizó un modelo entrenado en el conjunto de datos de ImageNet Large Visual Recognition Challenge. Estos modelos pueden diferenciar entre 1,000 clases diferentes. Tendrá la opción de elegir entre las arquitecturas de modelo, para que pueda determinar el equilibrio correcto entre la velocidad, el tamaño y la precisión de su problema.

3.7.1 MODELO DE RED NEURONAL CONVOLUCIONAL

Existen diferentes arquitecturas para redes neuronales, en particular, el presente trabajo de grado utilizará la arquitectura conocida como Inception v3 la cual fue propuesta en el artículo titulado “Rethinking the Inception Architecture for Computer Vision” (Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z., 2016).

Esta arquitectura consiste en una red neural profunda con muchos niveles en los cuales se realizan muchas de las operaciones que normalmente se encuentran en redes neuronales convolucionales más sencillas como lo es la capa conv, la capa que realiza el promedio sobre una región de los elementos de la capa anterior, la capa completamente conectada, etc.

Esta gran cantidad de capas permite a este modelo obtener una exactitud mayor que otras redes neuronales más sencillas. La figura 3.3 muestra la arquitectura de esta red neuronal

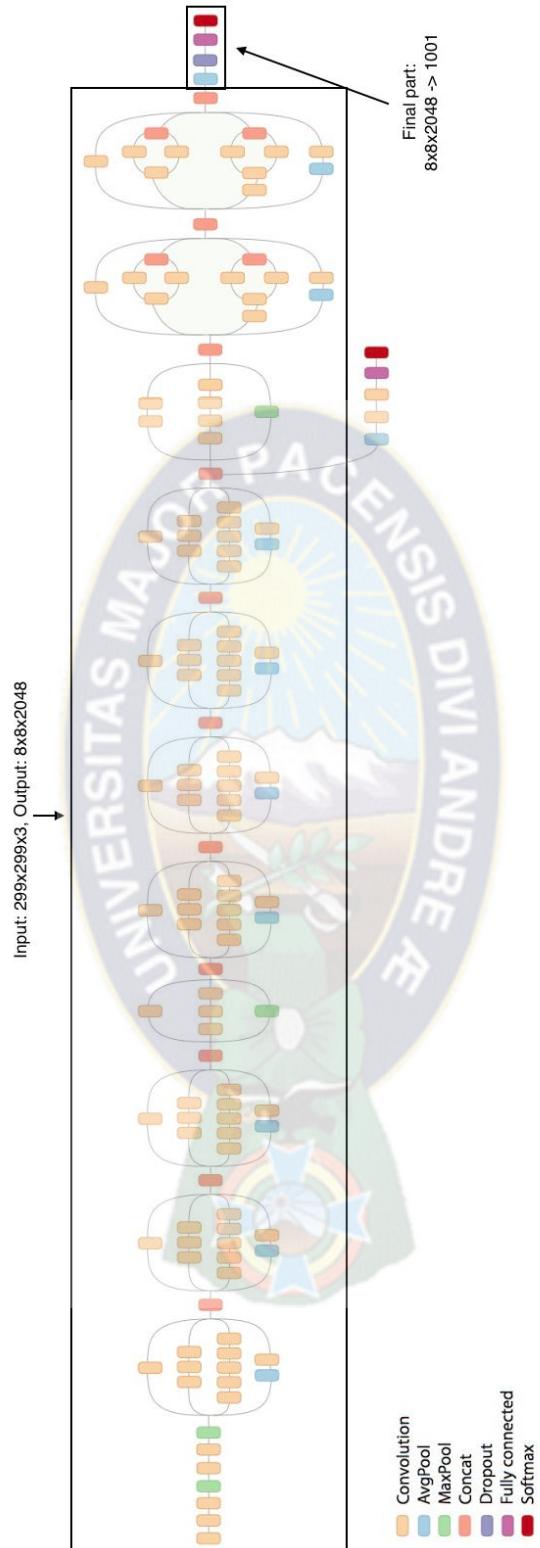


Figura 3.3: Arquitectura Inception v3.

Fuente: (Inception v3, 2015)

3.7.1.1 COMPONENTES PRINCIPALES DE LA RED NEURONAL CONVOLUCIONAL

Para crear nuestro modelo de red neuronal convolucional construiremos los cuatro componentes principales que conforman una red neuronal convolucional.

CAPA CONVOLUCIONAL

Consiste en un conjunto de filtros entrenables que realizan producto punto con los valores de la capa precedente. Los valores de los filtros son aprendidos para su activación al encontrar ciertas características. Se colocarán en forma de cascada para obtener diferentes niveles de abstracción.

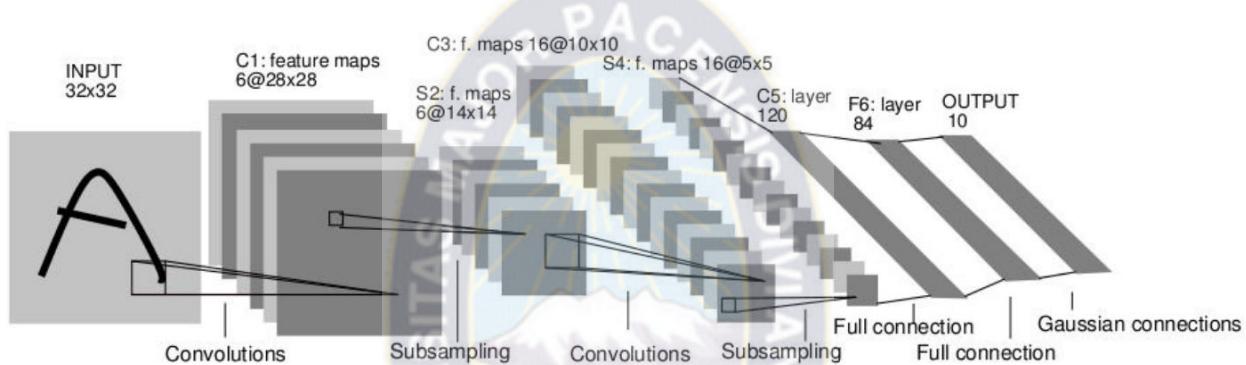


Figura 3.4: Capas convolucionales

Fuente: (Liba, 2016)

RECTIFICADOR LINEAL DE UNIDAD

Vamos a reemplazar los valores negativos por cero después de cada convolución y su propósito es agregar no linealidad al modelo, eliminando la relación proporcional entre la entrada y la salida.

POOLING

Sera el algoritmo utilizado para la reducción de la dimensión, con el objetivo de disminuir los tiempos de procesado reteniendo la información más importante.

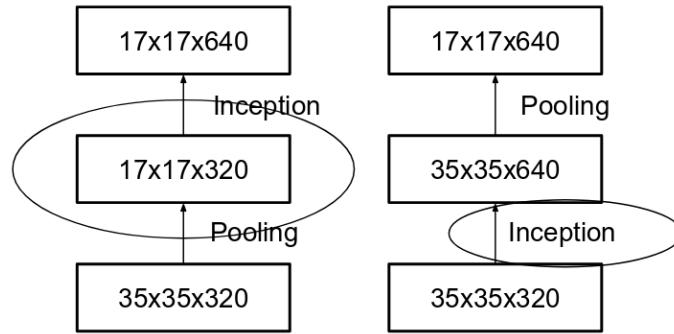


Figura 3.5: Pooling

Fuente: (Liba, 2016)

CAPA TOTALMENTE CONECTADA

Realiza la clasificación basado en las características extraídas por las capas de convolución y las reducidas por pooling. En esta capa todos los nodos están conectados con la capa precedente.

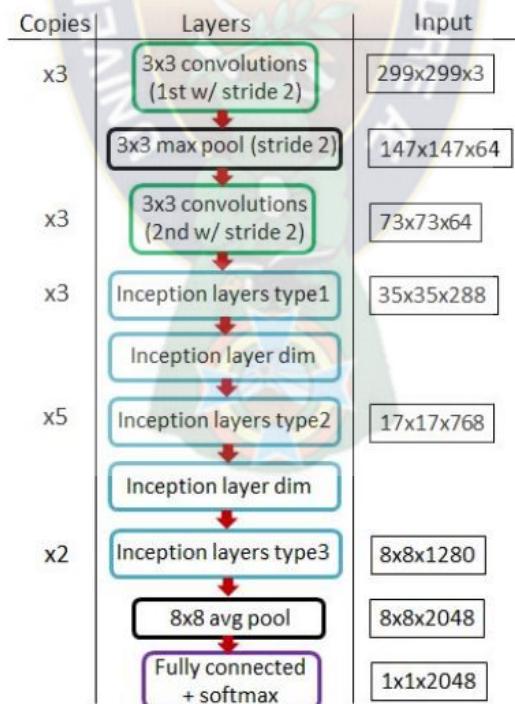


Figura 3.6: Capas totalmente conectada

Fuente: (Liba, 2016)

3.7.1.2 FUNCIONAMIENTO DE LA RED CONVOLUCIONAL

3.7.1.3 TRANSFERENCIA DE APRENDIZAJE

Para que una red neuronal pueda realizar la clasificación de las imágenes primero es necesario entrenarla, para ello se requiere de un conjunto de datos conformado por diferentes imágenes que correspondan con las clases o categorías que se requieren identificar, dicho conjunto de datos debe ser relativamente grande de manera que la red neuronal tenga suficientes muestras para aprender los patrones que caracterizan a cada categoría. Se cuenta con 6000 imágenes para cada letra del alfabeto, considerando que son 24 señas a ser reconocidas e interpretadas, en total se tiene un dataset de 144000 imágenes.

Para que la red neuronal pueda obtener mejores resultados al momento de realizar la clasificación de imágenes es necesario que las imágenes que se utilicen para su entrenamiento sean variadas, es decir, que muestren al objeto que se desea clasificar en diferentes condiciones de iluminación, diferentes ubicaciones y en diferentes orientaciones.

Para la recolección del dataset se contó con la participación de 10 personas de las cuales se extrajeron distintas imágenes de las representaciones de letras del alfabeto dactilológico en distintas condiciones de luz y distancia.

Debido a que se trata de un método de aprendizaje supervisado, además de las imágenes el sistema requiere que cada elemento tenga una etiqueta asociada a cada uno de ellos indicando a qué categoría pertenecen.

Lo anterior se realiza con la finalidad de que la red neuronal aprenda a identificar las características que son relevantes para la identificación del objeto y descartar aquellas que no lo son.

Tensorflow proporciona un algoritmo que permite asociar las imágenes con su respectiva etiqueta, pero primero se requiere que los archivos se encuentren organizados de cierta manera, manteniendo la siguiente estructura:

dataset/A/imagen1.jpg

dataset/A/imagen2.jpg

dataset/A/imagen3.jpg

dataset/B/imagen1.jpg

dataset/B/imagen2.jpg

dataset/C/imagen1.jpg

dataset/C/imagen2.jpg

dataset/C/imagen3.jpg

La etiqueta que el algoritmo asignará a cada imagen estará determinada por el nombre del folder en el cual se encuentra. Los nombres de cada archivo no son relevantes para este algoritmo.

Se hace uso del algoritmo denominado “retrain.py” para realizar el reentrenamiento de la red neuronal, este algoritmo se encuentra disponible en el repositorio de Tensorflow. (TensorFlow, 2018).

Antes de ejecutar el algoritmo es necesario indicar algunos parámetros que permiten ajustar el proceso para el tipo de hardware con el que se cuenta además de especificar algunas operaciones que se pueden realizar sobre las imágenes antes de que estas sean procesadas, esto con el fin de generar un conjunto de datos más variado, en particular se utilizaron los siguientes valores:

- ✓ how_many_training_steps = 12000
- ✓ train_batch_size = 12
- ✓ random_brightness = 150
- ✓ flip_left_right = True
- ✓ random_scale = 30

El parámetro “how_many_training_steps” indica la cantidad de pasos que el algoritmo debe ejecutar para realizar el reentrenamiento de la red.

El parámetro “train_batch_size” indica la cantidad de elementos que deben ser utilizados para realizar el reentrenamiento de la red en cada pasada, este valor debe ser modificado considerando la cantidad de memoria con la que se cuenta.

El parámetro “random_brightness” es un porcentaje indicando el rango aleatorio de valores por los que los pixeles de la imagen de entrada deben ser multiplicados.

El parámetro “flip_left_right” indica si algunas de las imágenes deben ser volteadas como si estuvieran reflejadas en un espejo.

El parámetro “random_scale” es un porcentaje indicando un rango aleatorio de valores para una escala que debe ser aplicada a las imágenes antes de que estas sean procesadas.

La figura 3.7 muestra parte de la salida que se obtiene durante el reentrenamiento de la red neuronal.

```
2017-05-06 20:32:26.168782: Step 8630: Train accuracy = 58.3%
2017-05-06 20:32:26.168848: Step 8630: Cross entropy = 1.463254
2017-05-06 20:32:29.011012: Step 8630: Validation accuracy = 75.4% (N=3188)
2017-05-06 20:32:34.992858: Step 8640: Train accuracy = 83.3%
2017-05-06 20:32:34.992926: Step 8640: Cross entropy = 0.846141
2017-05-06 20:32:37.892440: Step 8640: Validation accuracy = 74.4% (N=3188)
2017-05-06 20:32:43.889484: Step 8650: Train accuracy = 75.0%
2017-05-06 20:32:43.889555: Step 8650: Cross entropy = 1.114956
2017-05-06 20:32:46.694983: Step 8650: Validation accuracy = 73.1% (N=3188)
2017-05-06 20:32:52.683983: Step 8660: Train accuracy = 83.3%
2017-05-06 20:32:52.684051: Step 8660: Cross entropy = 0.777754
2017-05-06 20:32:55.548811: Step 8660: Validation accuracy = 74.2% (N=3188)
2017-05-06 20:33:01.502098: Step 8670: Train accuracy = 58.3%
2017-05-06 20:33:01.502168: Step 8670: Cross entropy = 1.065355
2017-05-06 20:33:04.459138: Step 8670: Validation accuracy = 74.2% (N=3188)
2017-05-06 20:33:10.463655: Step 8680: Train accuracy = 83.3%
2017-05-06 20:33:10.463725: Step 8680: Cross entropy = 0.727506
2017-05-06 20:33:13.270538: Step 8680: Validation accuracy = 73.9% (N=3188)
2017-05-06 20:33:19.294786: Step 8690: Train accuracy = 66.7%
2017-05-06 20:33:19.294859: Step 8690: Cross entropy = 1.192508
2017-05-06 20:33:22.169715: Step 8690: Validation accuracy = 73.8% (N=3188)
2017-05-06 20:33:28.373374: Step 8700: Train accuracy = 83.3%
2017-05-06 20:33:28.373451: Step 8700: Cross entropy = 0.871566
```

Figura 3.7: Reentrenamiento de la red neuronal.

Fuente: Elaboración propia.

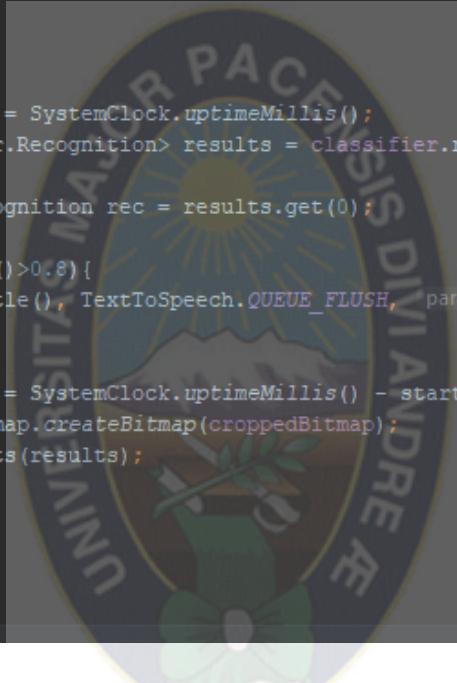
Una vez que el algoritmo concluye con el reentrenamiento de la red neuronal se generan dos archivos, el primero, un archivo de texto el cual contiene las etiquetas correspondientes a cada una de las categorías que la red es capaz de detectar. El segundo archivo generado contiene los nuevos valores de la red neuronal, estos serán utilizados para realizar la clasificación de nuevas imágenes.

3.8 INTERPRETACIÓN

Para realizar la traducción de la seña, realizada por la mano, al lenguaje oral accedemos a funciones de sonido, propias de Android, las que se encargan de la lectura y reproducción de sonido.

Se utilizó la API text-to-speech de android implementando un sintetizador de voz para la reproducción de la correspondiente traducción de la letra, esta fue implementada en la clase

ClasifierActivity del proyecto como se muestra en la figura, realiza la reproducción de voz solamente cuando la probabilidad de reconocimiento sea mayor a un 80 %.



```
if (SAVE_PREVIEW_BITMAP) {
    ImageUtils.saveBitmap(croppedBitmap);
}

tl=new TextToSpeech(getApplicationContext(), (status) -> {
    if(status != TextToSpeech.ERROR) {
        tl.setLanguage(Locale.getDefault());
    }
});

runInBackground(
    () -> {
        final long startTime = SystemClock.uptimeMillis();
        final List<Classifier.Recognition> results = classifier.recognizeImage(croppedBitmap);

        final Classifier.Recognition rec = results.get(0);

        if(rec.getConfidence()>0.8){
            tl.speak(rec.getTitle(), TextToSpeech.QUEUE_FLUSH, params: null);
        }

        lastProcessingTimeMs = SystemClock.uptimeMillis() - startTime;
        cropCopyBitmap = Bitmap.createBitmap(croppedBitmap);
        resultsView.setResults(results);
        requestRender();
        computing = false;
    });
}
```

Figura 3.8: Primera sección de la aplicación.

Fuente: Elaboración propia.

3.9. APLICACIÓN MÓVIL

Para que el usuario final pueda hacer uso de la red neuronal previamente entrenada para identificar imágenes este requiere de una plataforma fácil de usar, en este caso, dicha plataforma consiste en una aplicación móvil que permite capturar imágenes con la cámara de un dispositivo móvil para posteriormente identificar a qué clase corresponde.

Para desarrollar una aplicación para el sistema operativo Android primero se debe contar con ciertos elementos, como son:

- ✓ JDK 7 (Java Development Kit).
- ✓ JRE 6 (Java Runtime Environment).
- ✓ Android SDK.
- ✓ Android NDK.
- ✓ Android Studio.
- ✓ Emulador.

3.9.1. OPTIMIZAR EL MODELO

Los dispositivos móviles tienen limitaciones importantes, por lo que vale la pena considerar cualquier procesamiento previo que se pueda hacer para reducir la huella de una aplicación.

3.9.1.1. LIBRERÍAS LIMITADAS EN DISPOSITIVOS MÓVILES

Una forma en que la biblioteca de TensorFlow se mantiene pequeña, para dispositivos móviles, solo admite el subconjunto de operaciones que se usan comúnmente durante la inferencia. Este es un enfoque razonable, ya que la capacitación rara vez se lleva a cabo en plataformas móviles. Del mismo modo, también excluye el soporte para operaciones con grandes dependencias externas. Puede ver la lista de operaciones compatibles en el archivo tensorflow / contrib / makefile / tf_op_files.txt.

Por defecto, la mayoría de los gráficos contienen operaciones de entrenamiento que la versión móvil de TensorFlow no admite. TensorFlow no cargará un gráfico que contenga una operación no admitida (incluso si la operación no admitida es irrelevante para la inferencia).

3.9.1.2. OPTIMIZAR PARA LA INFERENCIA

Para evitar problemas causados por operaciones de capacitación no admitidas, la instalación de TensorFlow incluye una herramienta, optimize_for_inference, que elimina todos los nodos que no son necesarios para un conjunto determinado de entradas y salidas.

El script también hace algunas otras optimizaciones que ayudan a acelerar el modelo, como la fusión de operaciones explícitas de normalización por lotes en los pesos convolucionales para reducir la cantidad de cálculos. Esto puede dar una velocidad del 30%, dependiendo del modelo de entrada. Así es como ejecuta el script:

```
python -m tensorflow.python.tools.optimize_for_inference \
--input=tf_files/retrained_graph.pb \
--output=tf_files/optimized_graph.pb \
```

```
--input_names="input" \
--output_names="final_result"
```

La ejecución de este script crea un nuevo archivo en `tf_files/optimized_graph.pb`.

3.9.1.3. VERIFICAR EL MODELO OPTIMIZADO

Para comprobar que `optimize_for_inference` no ha alterado la salida de la red, compare la salida `label_image` para volver a `entrenar_graph.pb` con la de `optimized_graph.pb`

3.9.2. CONFIGURACIÓN DE LA APLICACIÓN ANDROID

La plataforma TensorFlow se encuentra escrita en el lenguaje C++ por lo que para poder ser utilizada por Android primero es necesario contar con una interfaz que permita a las aplicaciones escritas en java interactuar con ella.

La figura 3.9 muestra la estructura de archivos utilizada para guardar el archivo `libtensorflow_inference.so`



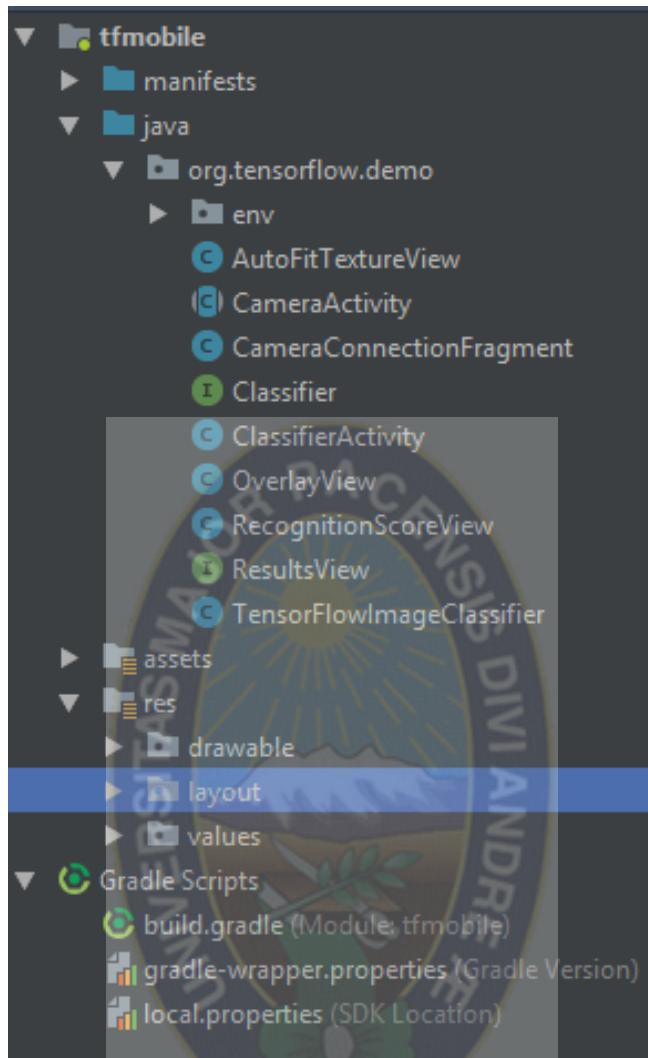


Figura 3.9: Estructura de archivos para Tensorflow en Android.

Fuente: Elaboración propia

Además de lo anterior también será necesario contar con un modelo, en este caso una red neuronal, previamente entrenado con el tipo de imágenes que se requiere identificar.

El modelo entrenado previamente contiene una serie de nodos que, si bien son útiles para realizar el entrenamiento de la red neuronal, no son requeridos para realizar la clasificación de las imágenes obtenidas por la aplicación, es por ello que se utiliza un script para removerlos (Tensorflow Android, 2017).

Esta operación tiene varias ventajas, como el hecho de reducir el tamaño del archivo correspondiente al modelo además de remover algunas operaciones que podrían no ser compatibles con ciertas plataformas.

Una vez que se han removido aquellos nodos innecesarios del modelo, se contará con dos archivos:

- ✓ output_graph.pb
- ✓ output_labels.txt

Estos archivos deberán ser colocados en la carpeta denominada assets dentro de la estructura del proyecto, tal y como se muestra en la figura 3.10.

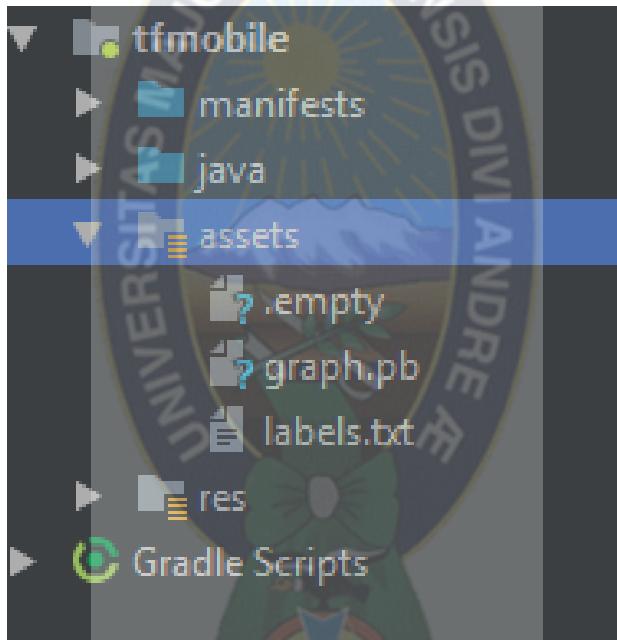


Figura 3.10: Estructura de archivos para el modelo previamente entrenado.

Fuente: Elaboración propia.

Una vez que se cuenta tanto con las bibliotecas de TensorFlow así como con el modelo previamente entrenado es necesario crear los elementos requeridos para interactuar con ellos y realizar la clasificación de las imágenes obtenidas.

3.10. DESCRIPCION DE LA APLICACIÓN MÓVIL

La aplicación realiza el reconocimiento e interpretación del alfabeto dactilológico de la lengua de señas, para ello muestra al usuario la imagen percibida por la cámara del dispositivo en ese momento.

Como se diseñó en la etapa de adquisición de imágenes, se tiene en la parte inferior la imagen percibida por la cámara del dispositivo móvil junto a una delimitación de área para el mejor enfoque de una seña en un rectángulo con márgenes.

La aplicación en primera instancia requiere de la autorización del permiso de uso de la cámara, el acceso a las imágenes, el contenido multimedia y los archivos del dispositivo tal y como se muestra en la figura 3.11.



Figura 3.11: Requerimiento de permisos de la aplicación

Fuente: Elaboración propia.

Dado el permiso, se desplegará la aplicación móvil para el reconocimiento e interpretación del alfabeto dactilológico de la lengua de señas como se muestra en la figura 3.12. Se encuadra la seña de la letra “a” en el rectángulo de la vista, se puede observar que en la parte superior nos muestra la asignación de la seña junto con su grado de probabilidad de un 0.9341843 y la reproducción de la voz sintetizada de la letra “a”.



Figura 3.12: Ejecución de la aplicación móvil

Fuente: Elaboración propia.

3.11. PRUEBAS

Para realizar el entrenamiento de la red neuronal que se encarga de clasificar las imágenes capturadas por la cámara del dispositivo móvil se utiliza un conjunto de imágenes de ejemplo y un

archivo de texto con las etiquetas correspondientes a cada una de las categorías que se desean identificar.

El conjunto de imágenes utilizado para el entrenamiento de la red neuronal es dividido en varios subconjuntos con el objetivo de ajustar los parámetros de la red y evaluar la calidad de los resultados generados, en este caso, los datos se dividen en:

- ✓ Datos de entrenamiento
- ✓ Datos de validación
- ✓ Datos de pruebas

El algoritmo utilizado para reentrenar la red neuronal genera una serie de registros que aportan información valiosa sobre el proceso.

Para poder visualizar estos registros, la plataforma Tensorflow proporciona una herramienta denominada Tensorboard la cual permite visualizar los datos generados en un navegador de internet.

Esta herramienta puede ser utilizada con el comando:

```
tensorboard logdir=/ruta/de/los/registros
```

Al invocar este comando se genera la salida mostrada en la figura 3.13, como se puede apreciar, la herramienta puede ser utilizada con cualquier navegador de internet, dirigiéndose a la dirección que se indica.

```
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library
libcublas.so.8.0 locally
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library
libcudnn.so.5 locally
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library
libcufft.so.8.0 locally
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library
libcuda.so.1 locally
I tensorflow/stream_executor/dso_loader.cc:135] successfully opened CUDA library
libcurand.so.8.0 locally
Starting TensorBoard b'41' on port 6006
(You can navigate to http://127.0.1.1:6006)
```

Figura 3.13: Herramienta Tensorboard

Fuente: Elaboración propia.

CAPÍTULO IV

ANÁLISIS DE RESULTADOS Y DEMOSTRACIÓN DE HIPOTESIS

4. INTRODUCCIÓN

En el presente capítulo se realiza el análisis de resultados obtenidos después de la fase de pruebas del anterior capítulo. Así también el desarrollo de la demostración de la hipótesis planteada en el primer capítulo del presente trabajo de grado.

4.1 ANÁLISIS DE DATOS

Una métrica útil para determinar el desempeño de la red neuronal es la exactitud con la que realiza la clasificación de los datos, es decir, la proporción de predicciones correctas de entre todas las predicciones realizadas por el sistema.

La figura 4.1 muestra la exactitud obtenida durante el proceso de reentrenamiento de la red neuronal con respecto al conjunto de datos de entrenamiento, representado por la línea naranja y el conjunto de datos de validación, representado por la línea verde.

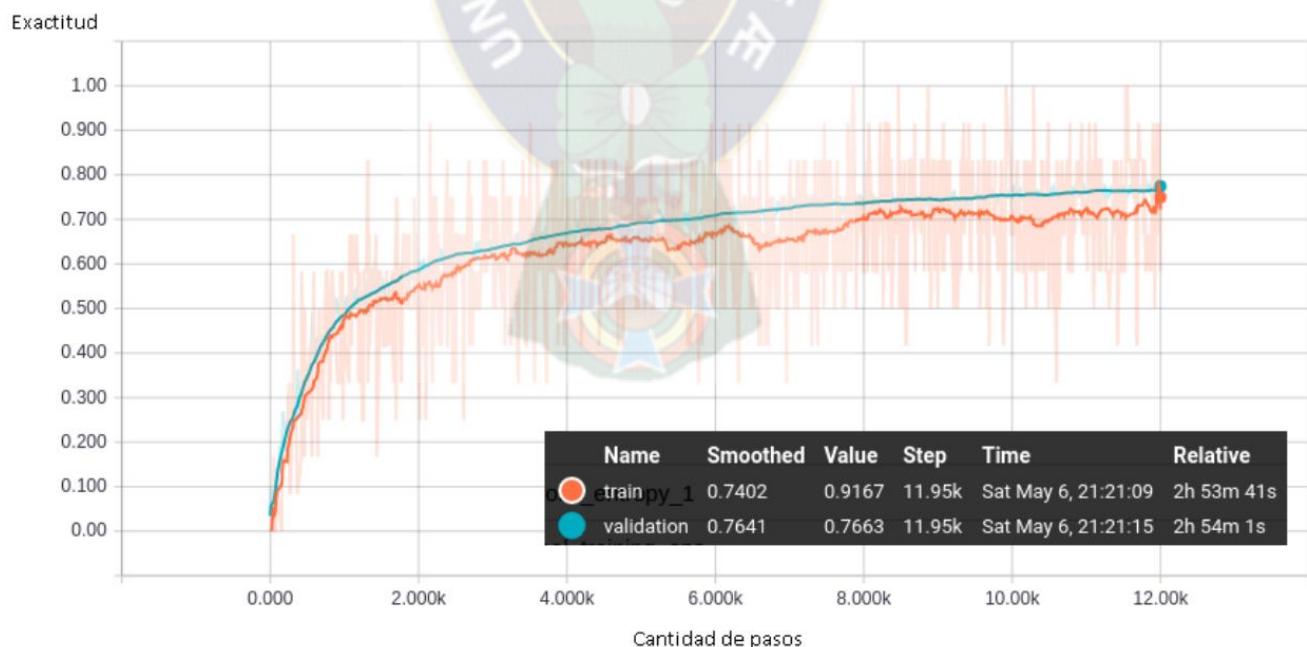


Figura 4.1: Exactitud de la red neuronal

Fuente: Elaboración propia.

Como se puede apreciar en un inicio la exactitud de la red neuronal se incrementa rápidamente con respecto al número de pasos, pero eventualmente se estabiliza y la exactitud se incrementa cada vez menos conforme aumenta el número de pasos.

Por lo tanto, es posible observar que el realizar una gran cantidad de pasos de entrenamiento no necesariamente resultará en un incremento significativo de la exactitud de la red neuronal.

Los experimentos realizados permitieron la construcción y la mejora del prototipo. De los experimentos se obtuvo lo siguiente.

- Uno de los factores con mayor influencia tiene para la interpretación de la señal es la iluminación del ambiente, ya que la inadecuada iluminación, distorsiona la calidad de la imagen con respecto al color de los objetos presentes en la captura o la no detección de dichos objetos.
- La distorsión ocasiona que la señal no sea debidamente reconocida, se puede apreciar la mala obtención de datos y la distorsión que causada por la falta de iluminación y excesiva iluminación respectivamente.
- Se redujo en un 1 % de confiabilidad del sistema al realizar la optimización para la implementación de la red neuronal artificial en dispositivos móviles.
- La distancia entre la cámara y la persona que realiza la dactilología de la lengua de señas no debe ser muy alejada, preferiblemente la mano debe estar en el cuadro de reconocimiento de la aplicación, de esta manera obtener mejores resultados.

4.2 DEMOSTRACIÓN DE LA HIPÓTESIS

La hipótesis que deseamos demostrar es la siguiente:

“El uso de redes neuronales artificiales permite realizar el reconocimiento e interpretación del alfabeto dactilológico de la lengua de señas en dispositivos móviles, con un grado de confiabilidad de al menos 90%”

4.2.1 CONTRASTE DE RACHAS DE WALD-WOLFOWITZ

Supongamos una población cuya función de distribución es desconocida y sea X la variable asociada a esa población, la cual solo puede tomar dos posibles valores como ejemplo, éxito (A) o fracaso (B).

H_0 : La muestra es aleatoria

H_1 : La muestra no es aleatoria

En general, sea una muestra de tamaño n en la que han aparecido n_1 elementos de tipo A y n_2 elementos del tipo B, siendo $n_1+n_2 = n$, y sea la variable aleatoria:

R = Número total de rachas en la muestra

Para una muestra grande y bajo la hipótesis H_0 es decir, para muestras aleatorias la distribución de probabilidad de R tiende hacia la normal a medida que n_1 y n_2 se van haciendo grandes. Esta aproximación es bastante buena si $n_1 > 10$ y $n_2 > 10$, de tal manera que:

$$R \rightarrow N(E[R], \sqrt{Var[R]})$$

Esperanza:

$$E[R] = \frac{2n_1 n_2}{n_1 + n_2} + 1$$

Varianza:

$$Var[R] = \frac{2n_1 n_2 (2n_1 n_2 - n_1 - n_2)}{(n_1 + n_2)^2 (n_1 + n_2 - 1)} + 1$$

Donde R es el número total de rachas observadas en la muestra. La región de aceptación para la hipótesis nula será:

$$-z_{\alpha/2} < z_{exp} < z_{\alpha/2}$$

El valor de $Z_{\alpha/2}$ se obtiene de la tabla de N (0,1), entonces:

$$P(z_1 \leq -z_{\alpha/2}) = P(Z_1 \geq z_{\alpha/2}) = \frac{\alpha}{2}$$

4.2.2 DESARROLLO DE LA DEMOSTRACIÓN DE LA HIPÓTESIS

Para el desarrollo de la demostración de la hipótesis por medio de contraste de rachas de Wald-Wolfowitz se siguen los siguientes pasos:

PASO 1: PLANTEAR HIPÓTESIS NULA E HIPÓTESIS ALTERNATIVA

H_0 = Las redes neuronales artificiales no permiten realizar el reconocimiento e interpretación del alfabeto dactilológico de la lengua de señas en dispositivos móviles.

H_i = El uso de redes neuronales artificiales permite realizar el reconocimiento e interpretación del alfabeto dactilológico de la lengua de señas en dispositivos móviles, con un grado de confiabilidad de al menos 90%

PASO 2: DETERMINAR NIVEL DE CONFIANZA

El nivel de confianza o significancia es de 90% lo que indica que $\alpha = 0.1$

PASO 3: IDENTIFICACIÓN DEL ESTADÍSTICO DE PRUEBA

En este caso se utiliza la prueba de rachas de Wald-Wolfowitz que utiliza los signos de los residuos y sus variaciones de negativo y positivo o viceversa. Una racha vendrá constituida por la sucesión de signos iguales.

PASO 4: FORMULACIÓN DE LAS REGLAS DE DECISIÓN

Para la prueba se toman 30 casos para el reconocimiento e interpretación de señas del alfabeto dactilológico, se realiza la comparación de identificación entre una persona y la aplicación móvil de reconocimiento e interpretación. A continuación, se muestra los resultados de comparación.

Nro.	Reconocimiento e interpretación por una persona	Reconocimiento e interpretación por la aplicación móvil	Aceptación por rachas
1	Reconocimiento de la letra “A”	Reconocimiento de la letra “A”	+
2	Reconocimiento de la letra “B”	Reconocimiento de la letra “A”	-
3	Reconocimiento de la letra “C”	Reconocimiento de la letra “C”	+
4	Reconocimiento de la letra “D”	Reconocimiento de la letra “D”	+

5	Reconocimiento de la letra “E”	Reconocimiento de la letra “A”	-
6	Reconocimiento de la letra “F”	Reconocimiento de la letra “A”	-
7	Reconocimiento de la letra “G”	Reconocimiento de la letra “G”	+
8	Reconocimiento de la letra “H”	Reconocimiento de la letra “H”	+
9	Reconocimiento de la letra “I”	Reconocimiento de la letra “I”	+
10	Reconocimiento de la letra “K”	Reconocimiento de la letra “Y”	-
11	Reconocimiento de la letra “L”	Reconocimiento de la letra “L”	+
12	Reconocimiento de la letra “M”	Reconocimiento de la letra “E”	-
13	Reconocimiento de la letra “N”	Reconocimiento de la letra “S”	-
14	Reconocimiento de la letra “O”	Reconocimiento de la letra “C”	-
15	Reconocimiento de la letra “P”	Reconocimiento de la letra “A”	-
16	Reconocimiento de la letra “Q”	Reconocimiento de la letra “Q”	+
17	Reconocimiento de la letra “R”	Reconocimiento de la letra “R”	+
18	Reconocimiento de la letra “S”	Reconocimiento de la letra “S”	+
19	Reconocimiento de la letra “T”	Reconocimiento de la letra “T”	+
20	Reconocimiento de la letra “U”	Reconocimiento de la letra “U”	+
21	Reconocimiento de la letra “V”	Reconocimiento de la letra “Y”	-
22	Reconocimiento de la letra “W”	Reconocimiento de la letra “Y”	-
23	Reconocimiento de la letra “X”	Reconocimiento de la letra “X”	+
24	Reconocimiento de la letra “Y”	Reconocimiento de la letra “A”	-
25	Reconocimiento de la letra “A”	Reconocimiento de la letra “A”	+
26	Reconocimiento de la letra “C”	Reconocimiento de la letra “C”	+
27	Reconocimiento de la letra “E”	Reconocimiento de la letra “E”	+
28	Reconocimiento de la letra “G”	Reconocimiento de la letra “G”	+
29	Reconocimiento de la letra “I”	Reconocimiento de la letra “Y”	-
30	Reconocimiento de la letra “K”	Reconocimiento de la letra “K”	+

Tabla 2.2: Comparación de reconocimiento e interpretación por una persona y la aplicación móvil

Fuente: (Vélez, et al., 2002).

Al hacer la evaluación con el sistema experto dieron los siguientes resultados:

$$(+)(-)(++)(--)(+++)(-)(+)(----)(+++++)(--)(+)(-)(+)$$

Donde:

(+) Representa los casos que la aplicación móvil reconoce e interpreta la señal.

(-) Los casos que la aplicación móvil no reconoce ni interpreta la señal.

Siendo racha construida por la sucesión de signos iguales se tiene que:

Total, de rachas: $R_{exp} = 15$

Número de observaciones: $N = 30$

Numero de residuos positivos: $n_1 = 18$

Numero de residuos positivos: $n_2 = 12$

Reemplazando datos para calcular la Esperanza y Varianza se tiene

Esperanza:

$$E[K] = \frac{2n_1 n_2}{n_1 + n_2} + 1 = \frac{2 * 18 * 12}{18 + 12} + 1 = \frac{432}{30} + 1 = 15,4$$

Varianza:

$$S^2[K] = \frac{2n_1 n_2 (2n_1 n_2 - n_1 - n_2)}{(n_1 + n_2)^2 (n_1 + n_2 - 1)} + 1 = \frac{2 * 18 * 12 (2 * 18 * 12 - 18 - 12)}{(18 + 12)^2 (18 + 12 - 1)} + 1$$

$$S^2[K] = \frac{173664}{26100} + 1 = 7,654 \rightarrow S(k) = \sqrt{7,654} = 2,15$$

PASO 5: TOMA DE DECISIÓN

Y para una muestra concreta el valor estadístico z_{exp} reemplazando datos se tiene:

$$z_{exp} = \frac{R - E[R]}{\sqrt{Var[R]}} = \frac{17 - 11,7}{\sqrt{2,15}} = \frac{5,3}{1,47} = 3,60$$

Calculando la región de aceptación de hipótesis, hallamos $z_{\alpha/2}$ que se muestra a continuación:

$$P\left(z_1 \leq -Z_{\alpha/2}\right) = P\left(Z_1 \geq Z_{\alpha/2}\right) = \frac{\alpha}{2}$$

Por lo tanto, la región de aceptación para la hipótesis nula es

$$-z_{\alpha/2} < z_{exp} < z_{\alpha/2}$$

$$-1,65 < z_{exp} < 1,65$$

Se puede ver que el estadístico $z_{exp} = -0,24$

Cae en el intervalo de aceptación, por lo tanto, se puede afirmar que la H_i : “El uso de redes neuronales artificiales permite realizar el reconocimiento e interpretación del alfabeto dactilológico de la lengua de señas en dispositivos móviles, con un grado de confiabilidad de al menos 90%” Lo que demuestra que la tesis es un trabajo valido, además muestra que los datos de la muestra son aleatorios.

Para la aceptación de la validez de la hipótesis planteada tenemos que recordar:

Que una vez que ha concluido el entrenamiento y el reentrenamiento de la red neuronal, es necesario verificar su exactitud con el conjunto de datos de pruebas, esta operación también es realizada por el algoritmo utilizado para el reentrenamiento de la red neuronal generando el resultado que se muestra en la figura 4.2.

```
and2_t_top_seg_1_croppedbne200.jpg.txt
Creating bottleneck at /home/administrador/Documents/trainedModel/bottleneck/t/h
and2_t_left_seg_4_cropped.jpg.txt
Creating bottleneck at /home/administrador/Documents/trainedModel/bottleneck/t/h
and2_t_top_seg_2_croppeddsw200.jpg.txt
Creating bottleneck at /home/administrador/Documents/trainedModel/bottleneck/t/h
and2_t_left_seg_5_croppednnw220.jpg.txt
Creating bottleneck at /home/administrador/Documents/trainedModel/bottleneck/t/h
and4_t_bot_seg_1_croppedrbc180.jpg.txt
Creating bottleneck at /home/administrador/Documents/trainedModel/bottleneck/t/h
and5_t_bot_seg_2_croppedrce200.jpg.txt
Creating bottleneck at /home/administrador/Documents/trainedModel/bottleneck/t/h
and3_t_dif_seg_4_cropped.jpg.txt
Creating bottleneck at /home/administrador/Documents/trainedModel/bottleneck/t/h
and3_t_dif_seg_2_croppednc250.jpg.txt
Creating bottleneck at /home/administrador/Documents/trainedModel/bottleneck/t/h
and1_t_top_seg_4_croppeddgn220.jpg.txt
Creating bottleneck at /home/administrador/Documents/trainedModel/bottleneck/t/h
and1_t_right_seg_3_croppednnw220.jpg.txt
Creating bottleneck at /home/administrador/Documents/trainedModel/bottleneck/t/h
and2_t_bot_seg_1_croppedrce200.jpg.txt
Final test accuracy = 77.6% (N=3106)
Converted 2 variables to const ops.
```

Figura 4.2: Exactitud final de la red neuronal

Fuente: Elaboración propia.

La red neuronal tiene una exactitud final de 77.6%, obtenida utilizando un conjunto de pruebas conformado por 144000 imágenes.

La exactitud no es mayor debido a que las imágenes utilizadas para resolver este problema son muy similares entre sí, todas las categorías consisten en imágenes de manos realizando diferentes señas por lo que las únicas diferencias entre una y otra son factores como la posición de los dedos o la orientación de la mano los cuales no generan diferencias lo suficientemente significativas, debido a esto la red neuronal puede llegar a confundir algunos de los signos, especialmente aquellos que son demasiado similares entre sí.



CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

4. CONCLUSIONES

La presente tesis de grado se ha desarrollado una aplicación móvil cuya finalidad es la de facilitar la comunicación para las personas sordas, permitiendo que cualquier persona sea capaz de entender el significado de algunas de las señas comúnmente utilizadas en las lenguas de señas.

La herramienta creada consiste en una aplicación móvil para el sistema operativo Android, tomando ventaja de que la gran mayoría de las personas cuentan con un dispositivo móvil y que de estos la mayoría cuenta con este sistema operativo además de que estos dispositivos cuentan con los todos los elementos de hardware que requiere la aplicación para su correcto funcionamiento.

Para realizar el análisis de las imágenes se utiliza una red neuronal profunda ya que este tipo de sistemas han demostrado buenos resultados para este tipo de tareas, sin embargo, se observa que debido a la complejidad de la red neuronal utilizada se requiere de una gran cantidad de tiempo, recursos de hardware y de datos para su correcto entrenamiento.

El problema anterior se resuelve utilizando la técnica de transferencia de aprendizaje la cual permite tomar una red neuronal previamente entrenada con una gran cantidad de datos y modificarla de tal manera que sea capaz de trabajar con un nuevo conjunto de datos.

También fue posible apreciar cómo a pesar de que los dispositivos móviles normalmente cuentan con una limitada capacidad de procesamiento estos son capaces de utilizar una red neuronal previamente entrenada para realizar la clasificación de imágenes.

Lo anterior es gracias a que, si bien es cierto que una red neuronal requiere de una cantidad considerable de recursos de hardware para su entrenamiento, el modelo generado al concluir el proceso no tiene requerimientos demasiado altos para su utilización.

Finalmente, la herramienta creada sirve para mostrar la manera de entrenar e integrar un sistema de inteligencia artificial (red neuronal profunda) con una aplicación móvil sin necesidad de utilizar un servidor remoto para realizar el análisis de los datos.

5. RECOMENDACIONES

El presente trabajo de grado estuvo enfocado en la solución del problema en dispositivos móviles con sistema operativo Android, pero sin embargo esta puede ser implementado en dispositivos móviles con sistema operativo iOS. Simplemente reutilizamos el modelo de red neuronal entrenado en el presente trabajo de grado para implementarlo en una aplicación iOS.

Para futuros trabajos en base a este modelo de red neuronal se recomienda incrementar el dataset de imágenes a una mayor cantidad para así poder incrementar el porcentaje de confiabilidad del sistema.

En la actualidad se conoce de técnicas de hibridación de redes neuronales artificiales con algoritmos genéticos, para poder plantear modelos de redes neuronales más eficientes se recomienda ampliar este trabajo hacia esas tendencias.



BIBLIOGRAFÍA

- Analytics10. (2018). *¿Cuál es la diferencia entre Inteligencia Artificial (AI) y Machine Learning (ML)?* Recuperado de: <https://www.analytics10.com/blog/cual-es-la-diferencia-entre-inteligencia-artificial-ai-y-machine-learning-ml/>.
- Android Developers. (2018). *Android Developers*. Recuperado de: <https://developer.android.com/>.
- Argollo Rafael, M. (2013). *Redes Neuronales para la interpretación del Lenguaje de Señas*. (Tesis de grado). Universidad Mayor de San Andrés, La Paz, Bolivia.
- Basterra, Berte, Botello, Castillo & Venturi. (2016) *Android OS Documentation Release 0.1*. Recuperado de: <https://media.readthedocs.org/pdf/androidos/latest/androidos.pdf>.
- Bronson. (2007). *C++ para ingeniería y ciencias*. mexico: Cengage learning Editores.
- Carhart R. (1947). *Hearing and Deafness*. Northwestern University: Holt, Rinehart, and Winston.
- Chacholla M. (2013). *Tutor inteligente para la enseñanza de la lectura y escritura para niños sordos*. (Tesis de grado). Universidad Mayor de San Andrés, La Paz, Bolivia.
- Comité Nacional Contra el Racismo y toda forma de Discriminación. (2014). *Poblacion y Cultura Sorda en Bolivia*. Recuperado de: <http://noracismo.gob.bo/archivos-pdf/sordos.pdf>.
- Del Brío M. & Sanz A. (1997). *Redes neuronales y sistemas borrosos*. Recuperado de: https://www.researchgate.net/publication/31732448_Redes_neuronales_y_sistemas_borrosos_B_Martin_del_Brio_A_Sanz_Molina_prol_de_Lotfi_A_Zadeh.
- EDMANS. (2006). *Técnicas y algoritmos básicos de Visión Artificial*. Universidad de La Rioja, España: Universidad de la Rioja.
- Fundación Adecco. (2017). *Tecnología y discapacidad*. Recuperado de <https://fundacionadecco.org/wp-content/uploads/2016/07/Informe-Tecnolog%C3%ADA-y-Discapacidad.-Fundaci%C3%B3n-Adecco-y-Keysight2017.pdf>.
- Hilera J. y Martinez V. (1995). *Redes neuronales artificiales: Fundamentos, modelos y aplicaciones*. Madrid: Ra-Ma.

Instituto Nacional de Estadística. (2012). *Censo Nacional de Población y Vivienda 2012*. Recuperado de: <http://censosbolivia.ine.gob.bo/webine/>.

Jaguar Labs. (2015). *Dilo en señas* (Versión 2.0) [Aplicación Móvil]. Descargado de: <https://play.google.com/store/apps/details?id=com.jaguarlabs.lsm>.

La Razón. (2017). *Una app traduce el lenguaje de señas*. Recuperado de: http://www.la-azon.com/suplementos/mia/app-traduce-lenguaje-senas_0_2786721321.html.

Lara Felipe. (2015). *Fundamentos de Redes Neuronales Artificiales*. Recuperado de: http://conceptos.sociales.unam.mx/conceptos_final/598trabajo.pdf.

Maduell, E. (2015). *Visión Artificial*. Recuperado de: [https://www.exabyteinformatica.com/uoc/Informatica/Diseno_de_interaccion/Diseno_de_interaccion_\(Modulo_5\).pdf](https://www.exabyteinformatica.com/uoc/Informatica/Diseno_de_interaccion/Diseno_de_interaccion_(Modulo_5).pdf).

Ministerio de Educación. (2016). *Discapacidad Auditiva*. Recuperado de: <http://www.minedu.gob.bo/ambitos/discapacidad/auditiva.html>.

Nightly-android Jenkins. (2017). *Tensorflow*. Recuperado de: <https://ci.tensorflow.org/view/Nightly/job/nightly-android/>

Olah Christopher. (2014). *Conv Nets: A Modular Perspective*. Recuperado de: <https://colah.github.io/posts/2014-07-Conv-Nets-Modular/>.

Organización mundial para la salud. (2011). Informe mundial sobre la discapacidad. Recuperado de: http://www.who.int/disabilities/world_report/2011/accessible_es.pdf?ua=1.

Palmer A., Montaño J., & Calafat A. (2000). Predicción del consumo de éxtasis a partir de redes neuronales artificiales. Recuperado de: http://www.irefrea.eu/uploads/PDF/Palmer%20et%20al_2000_Prediccion%20consumo%20extasis.pdf.

Platero C. (2009). *Apuntes de Visión Artificial*. Recuperado de: http://www.elai.upm.es/webantigua/spain/Asignaturas/MIP_VisionArtificial/ApuntesVA/cap1IntroVA.pdf.

Ponce Pedro C. (2010). *Inteligencia Artificial con aplicaciones a la ingeniería*. Mexico D.F., Mexico: Alfaomega Grupo Editor, S.A. de C.V.

Samsung (2016). *Especificaciones Samsung Galaxy 2016*. Recuperado de:
<http://www.samsung.com/es/smartphones/galaxy-j5-j500fn/SM-J500FZWAPHE/>.

SIGMA. (2017). *Inteligencia Artificial*. Recuperado de:
<http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/219/A7.pdf>

Signlab. (2013). *Lengua de Signos textoSIGN* (Versión 2.0) [Aplicación Móvil]. Descargado de:
https://play.google.com/store/apps/details?id=lse.full&hl=es_419.

Statcounter. (2018). *Operating System Market Share Worldwide*. Recuperado de
<http://gs.statcounter.com/os-market-share>.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). *Rethinking the inception architecture for computer vision*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2818-2826).

TensorFlow. (2018). *Developer Tutorial*. Recuperado de : <https://www.tensorflow.org/tutorials/>.

TensorFlow Android. (2017). *GitHub*. Recuperado de:
<https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android>.

TBWA\España, Asociación para la Normalización del Lenguaje de Signos. (2014). *Signslator*. Recuperado de: <http://www.signslator.com/>.

University of Reading. (2008). *Using digital images in teaching and learning*. Recuperado de
<https://www.reading.ac.uk/web/files/using-images/Understanding1.pdf>.

Valiente P. (2016). *TensorFlow, tutorial básico*. Recuperado de:
<http://www.p.valienteverde.com/tensorflow-tutorial-basico/>.

Vázquez R. A. (2008). *A new associative model with dynamical synapses*, “*Neural Processing Letters*” vol. 28, no. 3, pp. 189–207, 2008.

Vélez J., Moreno A., Sánchez A., & Sanchez J. (2002). *Visión por computador*. Recuperado de:
<http://www.visionporcomputador.es/libroVision/VisionPorComputador.pdf>.

Vilches J. (2005). *La dactilología, ¿qué, cómo, cuándo...?*. Recuperado de:
http://www.uco.es/~fe1vivim/alfabeto_dactilogico.pdf.

OpenCV team. (2018). *About*. Recuperado de: <https://opencv.org/about.html>.

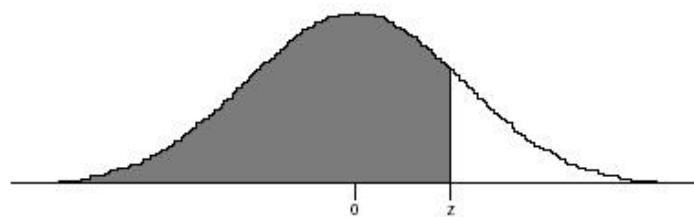




ANEXOS

ANEXO A

TABLA NORMAL DE DISTRIBUCIÓN

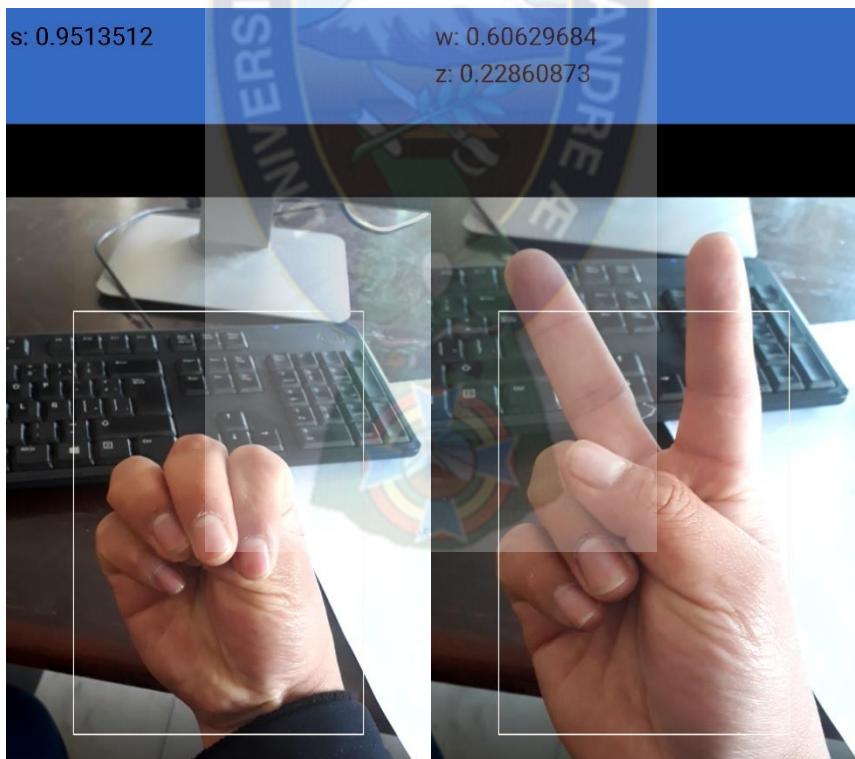


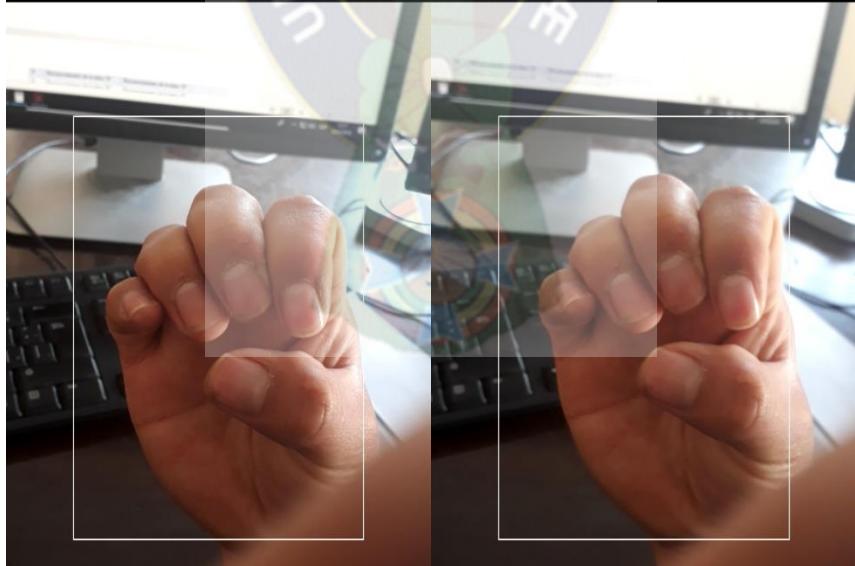
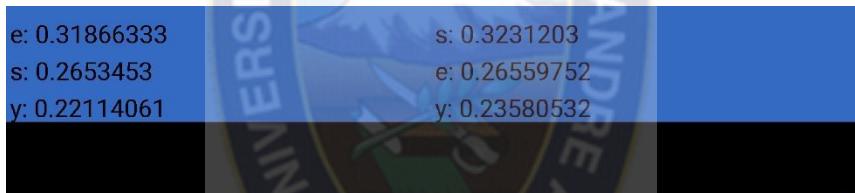
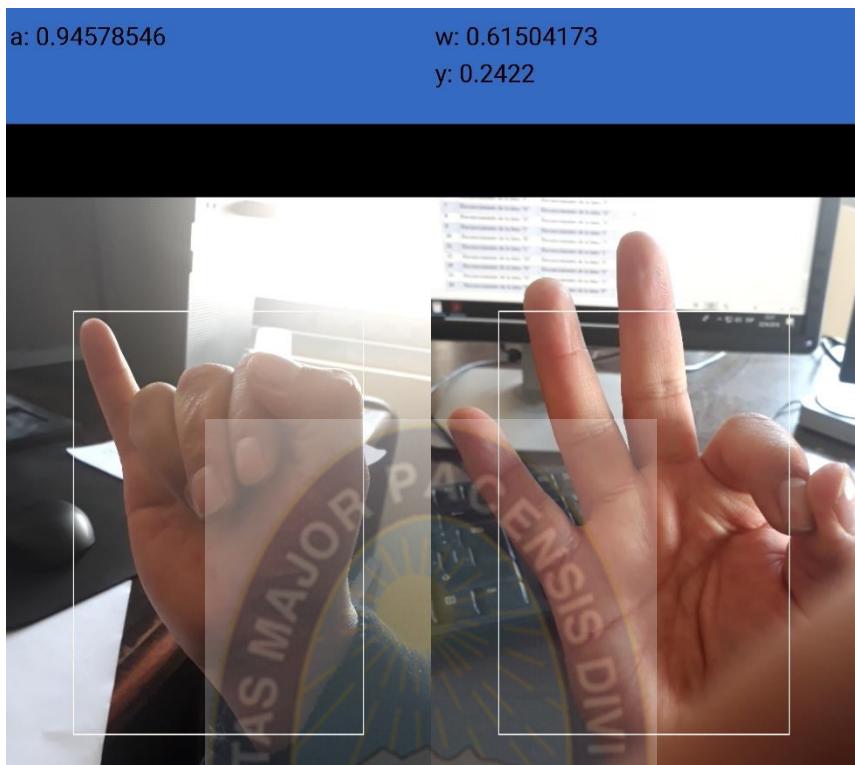
Normal Deviate z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
-4.0	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
-3.9	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
-3.8	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
-3.7	.0001	.0001	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
-3.6	.0002	.0002	.0001	.0001	.0001	.0001	.0001	.0001	.0001	.0001
-3.5	.0002	.0002	.0002	.0002	.0002	.0002	.0002	.0002	.0002	.0002
-3.4	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0002
-3.3	.0005	.0005	.0005	.0004	.0004	.0004	.0004	.0004	.0004	.0003
-3.2	.0007	.0007	.0006	.0006	.0006	.0006	.0006	.0005	.0005	.0005
-3.1	.0010	.0009	.0009	.0009	.0008	.0008	.0008	.0008	.0007	.0007
-3.0	.0013	.0013	.0013	.0012	.0012	.0011	.0011	.0011	.0010	.0010
-2.9	.0019	.0018	.0018	.0017	.0016	.0016	.0015	.0015	.0014	.0014
-2.8	.0026	.0025	.0024	.0023	.0023	.0022	.0021	.0021	.0020	.0019
-2.7	.0035	.0034	.0033	.0032	.0031	.0030	.0029	.0028	.0027	.0026
-2.6	.0047	.0045	.0044	.0043	.0041	.0040	.0039	.0038	.0037	.0036
-2.5	.0062	.0060	.0059	.0057	.0055	.0054	.0052	.0051	.0049	.0048
-2.4	.0082	.0080	.0078	.0075	.0073	.0071	.0069	.0068	.0066	.0064
-2.3	.0107	.0104	.0102	.0099	.0096	.0094	.0091	.0089	.0087	.0084
-2.2	.0139	.0136	.0132	.0129	.0125	.0122	.0119	.0116	.0113	.0110
-2.1	.0179	.0174	.0170	.0166	.0162	.0158	.0154	.0150	.0146	.0143
-2.0	.0228	.0222	.0217	.0212	.0207	.0202	.0197	.0192	.0188	.0183
-1.9	.0287	.0281	.0274	.0268	.0262	.0256	.0250	.0244	.0239	.0233
-1.8	.0359	.0351	.0344	.0336	.0329	.0322	.0314	.0307	.0301	.0294
-1.7	.0446	.0436	.0427	.0418	.0409	.0401	.0392	.0384	.0375	.0367
-1.6	.0548	.0537	.0526	.0516	.0505	.0495	.0485	.0475	.0465	.0455
-1.5	.0668	.0655	.0643	.0630	.0618	.0606	.0594	.0582	.0571	.0559
-1.4	.0808	.0793	.0778	.0764	.0749	.0735	.0721	.0708	.0694	.0681
-1.3	.0968	.0951	.0934	.0918	.0901	.0885	.0869	.0853	.0838	.0823
-1.2	.1151	.1131	.1112	.1093	.1075	.1056	.1038	.1020	.1003	.0985
-1.1	.1357	.1335	.1314	.1292	.1271	.1251	.1230	.1210	.1190	.1170
-1.0	.1587	.1562	.1539	.1515	.1492	.1469	.1446	.1423	.1401	.1379
-.9	.1841	.1814	.1788	.1762	.1736	.1711	.1685	.1660	.1635	.1611
-.8	.2119	.2090	.2061	.2033	.2005	.1977	.1949	.1922	.1894	.1867
-.7	.2420	.2389	.2358	.2327	.2296	.2266	.2236	.2206	.2177	.2148
-.6	.2743	.2709	.2676	.2643	.2611	.2578	.2546	.2514	.2483	.2451
-.5	.3085	.3050	.3015	.2981	.2946	.2912	.2877	.2843	.2810	.2776
-.4	.3446	.3409	.3372	.3336	.3300	.3264	.3228	.3192	.3156	.3121
-.3	.3821	.3783	.3745	.3707	.3669	.3632	.3594	.3557	.3520	.3483

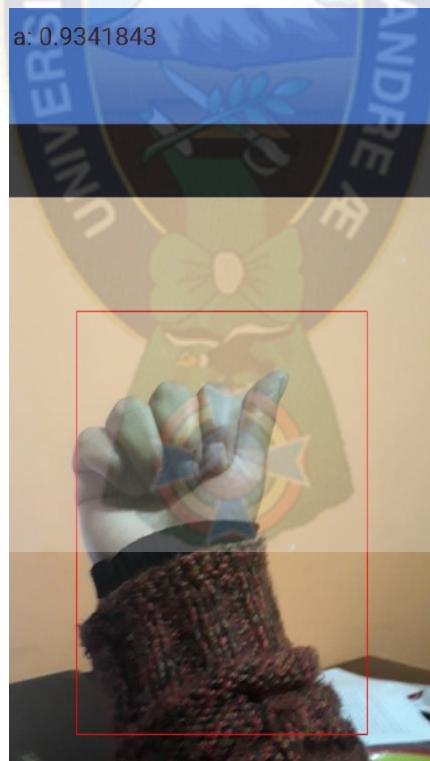
ANEXO B

PRUEBAS DE LA APLICACIÓN MÓVIL











DOCUMENTACIÓN