UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

RECONOCIMIENTO AUTOMÁTICO DE LA SIMBOLOGÍA BASE DE LA ESCRITURA INCA UTILIZANDO MODELOS DEEP CONVOLUTIONAL NEURAL NETWORK

PRESENTADA POR:

LENIN HUAYTA FLORES

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

PUNO, PERÚ

2024

Reporte de similitud

NOMBRE DEL TRABAJO

RECONOCIMIENTO AUTOMÁTICO DE LA SIMBOLOGÍA BASE DE LA ESCRITURA IN CA UTILIZANDO MODELOS DEEP CONVO LUTIONAL NEURAL NETWORK

AUTOR

LENIN HUAYTA FLORES

RECUENTO DE PALABRAS

26085 Words

RECUENTO DE CARACTERES

154615 Characters

RECUENTO DE PÁGINAS

118 Pages

TAMAÑO DEL ARCHIVO

1.9MB

FECHA DE ENTREGA

Sep 10, 2024 1:39 PM GMT-5

FECHA DEL INFORME

Sep 10, 2024 1:40 PM GMT-5

8% de similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para cada base de datos.

- 5% Base de datos de Internet
- 1% Base de datos de publicaciones
- Base de datos de Crossref
- Base de datos de contenido publicado de Crossref
- 6% Base de datos de trabajos entregados

Excluir del Reporte de Similitud

- Material bibliográfico
- Material citado
- Material citado
- Coincidencia baja (menos de 10 palabras)

Dr. Edgar Eloy Carpio Vargas
INGENIERO ESTADÍSTICO

UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO - PERÚ
ESCUELA DE POSGRADO
COORDINACIÓN DE INVESTIGACIÓN

Ruben Jared Luque Coyla
ING. ESTADÍSTICO E INFORMÁTICO
CIP. 116625

Resumen

UNIVERSIDAD NACIONAL DEL ALTIPLANO

ESCUELA DE POSGRADO

DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN

TESIS

RECONOCIMIENTO AUTOMÁTICO DE LA SIMBOLOGÍA BASE DE LA ESCRITURA INCA UTILIZANDO MODELOS DEEP CONVOLUTIONAL NEURAL NETWORK

PRESENTADA POR:

LENIN HUAYTA FLORES

PARA OPTAR EL GRADO ACADÉMICO DE:

DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

APROBADA POR EL JURADO SIGUIENTE:

PRESIDENTE

.....
Dr. BERNABE CANQUI FLORES

PRIMER MIEMBRO

.....
D.Sc. PERCY HUATA PANCA

SEGUNDO MIEMBRO

.....
Dr. MILTON ANTONIO LOPEZ CUEVA

ASESOR DE TESIS

.....
Dr. EDGAR ELOY CARPIO VARGAS

Puno, 18 de julio de 2024.

ÁREA: Redes y comunicaciones.
TEMA: Reconocimiento automático de la simbología base de la escritura Inca utilizando modelos Deep Convolutional Neural Network.
LÍNEA: Desarrollo de aplicaciones.

DEDICATORIA

A mis ancestros de los Andes de Sudamérica, a toda mi familia; en especial a mi compañera de vida Dania Yin y a mis hijos Lenin Samir y Yang Lenin.

A los investigadores, inventores, innovadores, emprendedores, desarrolladores, científicos, académicos, profesionales, trabajadores, estudiantes y público en general.

Lenin Huayta Flores.

AGRADECIMIENTOS

A mi alma mater Universidad Nacional del Altiplano, muy en particular al doctorado en Ciencias de la Computación por darme esta oportunidad de estudiar.

A todo el personal docente, administrativo y estudiantil de la Universidad Nacional del Altiplano que apoyaron directa e indirectamente en la conclusión de esta tesis.

Lenin Huayta Flores.

ÍNDICE GENERAL

CAPÍTULO III

MATERIALES Y MÉTODOS

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

ACRÓNIMOS

ADAM	:	*Adaptive Moment Estimation*
AHCD	:	*Arabic Handwritten Characters Dataset*
ARDIS	:	*Arkiv Digital Sweden*
CAMShift	:	*Continuously Adaptive Mean Shift*
CASIA	:	*Institute of Automation of Chinese Academy of Sciences*
CEDaR	:	*Centre for Environmental Data and Recording*
CNN	:	*Convolutional Neural Network*
CPU	:	*Central Processing Unit*
DCNN	:	*Deep Convolutional Neural Network*
FN	:	Falso Negativo
FP	:	Falso Positivo
FT	:	*Fourier Transform*
GB	:	*Gigabyte*
GPU	:	*Graphics Processing Unit*
HBCL	:	*Handwritten Balinese Characters of Lontar*
HWDB	:	*Chinese Hand-Written DataBase*
ICDAR	:	*International Conference on Document Analysis and Recognition*
IFN/ENIT	:	*Institut für Nachrichtentechnik/ Ecole Nationale d'Ingénieurs de Tunis*
ILSVRC	:	*ImageNet Large Scale Visual Recognition Challenge*
MNIST	:	*Modified National Institute of Standards and Technology*
OHASD	:	*Online Handwritten Arabic Sentence Database*
RAM	:	*Random Access Memory*
RGB	:	*Red, Green, Blue*
ReLU	:	*Rectified Lineal Unit*
SGD	:	*Stochastic Gradient Descent*
VGG	:	*Visual Geometry Group*
VN	:	Verdadero Negativo
VP	:	Verdadero Positivo

RESUMEN

A pesar de los avances en la interpretación numérica de los quipus, una comprensión integral de los quipus de escritura, especialmente aquellos del estadío Inca, sigue siendo limitada. Este estudio identifica patrones utilizando algoritmos de procesamiento de imágenes y visión por computadora; el objetivo fue desarrollar un sistema de red neuronal convolucional profunda que obtenga mejores resultados en el reconocimiento automático de la simbología base de la escritura Inca; en cuanto al método, la investigación fue aplicada, explicativa, cuantitativa, longitudinal y se enmarcó en el paradigma positivista; la población estuvo conformada por 167 clases o palabras/frases y la muestra estuvo conformada por 40 clases o palabras/frases; como resultado, se recolectaron, estructuraron y almacenaron símbolos provenientes de quipus de escritura, escritura sobre madera y tejidos; se creó un conjunto de datos a partir de los símbolos identificados y etiquetados, del cual el 80% de las imágenes se utilizaron para entrenamiento y el 20% para validación; se evaluaron cinco técnicas de redes neuronales convolucionales profundas entrenadas mediante aprendizaje por transferencia bajo las mismas condiciones y métricas, y *DenseNet121* obtuvo el mejor resultado en la detección de la simbología base, con una exactitud del 98,63% en la validación. Concluimos que el sistema obtuvo resultados superiores en el reconocimiento automático de la simbología base de los quipus de escritura, superando la precisión de otros métodos; estos hallazgos validan la efectividad del modelo y respaldan el uso del método propuesto.

Palabras clave: Aprendizaje por transferencia, idioma quechua, modelo preentrenado, quipus de escritura, reconocimiento automático de escritura, red neuronal convolucional profunda.

ABSTRACT

Despite the advances in the numerical interpretation of quipus, a comprehensive understanding of quipus used for writing, particularly those from the Inca period, remains limited. This study identifies patterns using image processing and computer vision algorithms; the objective was to develop a deep convolutional neural network system that obtains better results in the automatic recognition of the base symbology of Inca writing; regarding the method, the research was applied, explanatory, quantitative, longitudinal and was framed in the positivist paradigm; the population was made up of 167 classes or words/phrases and the sample was made up of 40 classes or words/phrases; as a result, symbols from writing quipus, writing on wood and fabrics were collected, structured and stored; a dataset was created from the identified and labeled symbols, of which 80% of the images were used for training and 20% for validation; five deep convolutional neural network techniques trained using transfer learning were evaluated under the same conditions and metrics, and DenseNet121 obtained the best result in detecting the base symbology, with an accuracy of 98.63% in the validation. We conclude that the system obtained superior results in the automatic recognition of the base symbology of the writing quipus, surpassing the precision of other methods; these findings validate the effectiveness of the model and support the use of the proposed method; these findings validate the effectiveness of the model and support the use of the proposed method.

Keywords: Automatic handwriting recognition, deep convolutional neural network, pre-trained model, Quechua language, quipus of writing, transfer learning.

INTRODUCCIÓN

Algunos investigadores precisan que los habitantes precolombinos de los Andes, no desarrollaron una escritura fonográfica, silábica o alfabética, por ausencia de transcripciones conservadas hasta nuestros tiempos; en contraposición en esta investigación se realiza un estudio que demostraría lo contrario de acuerdo a los estudios de De la Jara (2017) y Laurencich-Minelli (2016a, 2016b), para lo cual en este estudio se recopila, estructura y almacena cada imagen o símbolo de la escritura. Una de las formas de escritura del estadío incaico fue la pictográfica (Silverman, 2011) estuvo perdida hasta 1962 (De la Jara, 2017) y consta de palabras primarias que tienen un significado y la separación por silabas (Roncoroni, 2017) que corresponden a la cantidad los nudos de los *qhapaq khipu teqsi simi* (Huillca-Huallpa, 2016) o *capacquipu* (Laurencich-Minelli, 2016a) y que es parte del quipu numérico (Laurencich-Minelli, 2016b); para su reconocimiento se identifica la palabra base o primaria en el quipu real, posteriormente se separa en sílabas de acuerdo a la cantidad de nudos en la cuerda; se realiza de manera iterativa con las siguientes palabras primarias, hasta formar la sílaba, palabra, o frase. Actualmente las redes neuronales convolucionales profundas permiten reconocer la escritura en diferentes idiomas como el árabe, polaco, chino (Altwaijry y Al-Turaiki, 2021; Lukasik et al., 2021) y dialectos; caracteres chinos y latinos; escrituras como manipuri, bengalí y meitei mayek; los investigadores plantean a futuro desarrollar una convolución más eficiente para una clasificación precisa de la escritura (Hazra et al., 2021; Rehman, 2021; Wang y Du, 2021) y la extracción de características que son resistentes a las rotaciones de la imagen en diferentes direcciones (Nanehkaran et al., 2021) y la escritura en quipus de la época incaica no está exenta de estos problemas y es necesario tener un conjunto de datos del mismo, entre símbolos y dígitos (Inunganbi et al., 2021), y si es necesario realizar la hibridación en la arquitectura de red neuronal convolucional profunda (*DCNN*) para reducir la complejidad del procesamiento (Zhuang et al., 2021). En este estudio se propone un modelo de reconocimiento automático de los quipus de escritura utilizando un algoritmo de visión computacional basado en una técnica *DCNN* (Hazra et al., 2021), para validar el rendimiento, se construirá un conjunto de datos a partir de los símbolos identificados y etiquetados (Abderrahim et al., 2020) para el reconocimiento óptico complejo de palabras y oraciones base (Mridha et al., 2021), y se obtienen resultados que exhiben una exactitud promedio mayor a 95%

(Elleuch et al., 2021; Khan et al., 2021) que nos permite reconocer automáticamente la simbología base de los quipus de escritura.

La investigación está organizada en capítulos secuenciales de la siguiente manera: El Capítulo I proporciona una revisión de la literatura relacionada con el estudio, cubriendo las bases teóricas y conceptuales. El Capítulo II se centra en el planteamiento del problema, que comprende la descripción, identificación y formulación del problema de investigación, así como la justificación, los objetivos y las hipótesis. En el Capítulo III se describen los materiales y métodos empleados, incluyendo el lugar de estudio, la población y la muestra seleccionadas, el método de investigación, y una descripción detallada de los métodos utilizados para alcanzar los objetivos específicos. El Capítulo IV expone los resultados y su discusión correspondiente. Finalmente, se presentan las conclusiones de la investigación, se ofrecen recomendaciones basadas en sugerencias útiles, y se incluye la bibliografía y los anexos.

CAPÍTULO I

REVISIÓN DE LITERATURA

1.1 Marco teórico

1.1.1 Simbología base de la escritura Inca

Destruido el horizonte cultural incaico y con eso los amautas, los quipus se han vuelto ininteligible porque ya no queda nadie que comparta esa cultura. Cuando los investigadores modernos tratan de construir alfabetos con los quipus, renuncian desde el comienzo a entender que es una escritura de un horizonte cultural totalmente diferente, que no puede ser reconducido a una cultura alfabética occidental (Rivera, 2021); para entenderlos hay un solo camino: conocer a la cultura incaica hasta ensimismarse en ella.

A. Idioma Inca

Generalmente hace referencia al quechua, una lengua que posee diversos dialectos. La forma estándar del quechua, conocida como Quechua II, se habla en la región central de los Andes. Este idioma cuenta con una rica tradición oral y cultural; y ha mantenido su vigencia a lo largo de los siglos a pesar de los cambios sociopolíticos en la región.

A.1 Quechua

Denominado también *quichua, qquichua* (Gonzalez-Holguin, 1608), *runa simi* (Ricardo, 1586) o *qhapaq khipu teqsi simi* (Huillca-Huallpa, 2016) es la lengua indígena más importante de las Américas (Carreño, 2006), específicamente de los Andes de Sudamérica. Se estima que actualmente entre 7 y 8 millones de personas hablan este idioma en varios países de Sudamérica. La gran mayoría está concentrada en los países de los andes centrales, Ecuador, Perú y Bolivia; mientras que bolsones menores de hablantes ubican al quechua también en Argentina, Colombia, Chile, y tal vez también en Brasil (Cerrón-Palomino, 1987; Hornberger y Coronel-Molina, 2004).

A.2 Quipu

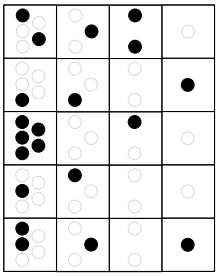
Los quipus son un sistema cuerdas anudadas con fines contables o mnemotécnicos. La palabra quipu proviene del quechua *quipu* y significa nudo (Ricardo, 1586). Generalmente, se ha considerado que los quipus estuvieron asociados a la cultura Inca, aunque algunos hallazgos fortuitos habían sugerido que su antigüedad podía remontarse más de medio milenio atrás (Shady et al., 2000). Se han encontrado quipus en Caral, la ciudad más antigua de América, y en la ciudad de Wari. El quipu más antiguo data del año 2500 a. n. e. y estaban confeccionados con algodón o con lana de llama o alpaca. Estos quipus eran teñidos y anudados. Una vez hecho los hilos en el caso de los quipus de contabilidad se codificaban en valores numéricos siguiendo un sistema posicional de base decimal (Ayala, 2017). Salomon (2004) en un estudio etnográfico acerca del quipu redefine al concepto de escritura como universo de lo legible se supera el obstáculo de definir con suficiente amplitud la idea de escritura, pero subordinándola a la de lectura.

A.3 *Yupana*

Es un ábaco para realizar cálculo matemático que fue utilizado por los *quipucamayos* como complemento del quipu en el imperio de los Incas (Moscovich, 2010).

Figura 1

Yupana



Nota. Adaptado de *Nueva Corónica y Buen Gobierno* (p. 360), por Poma (1615).

Según Poma (1615) los *quipucamayoc* contaban comenzando del cero [*ch'usaq*], uno [*huk*], dos [*iskay*], tres [*kimsa*], cuatro [*tawa*], 5

[*pisqa*], 6 [*suqta*], 7 [*qanchis*], 8 [*pusaj*], 9 [*isqun*], 10 [*chunka*], 20 [*iskay chunka*], 30 [*kimsa chunka*], 40 [*tawa chunka*], 50 [*pisqa chunka*], 60 [*suqta chunka*], 70 [*qanchis chunka*], 80 [*pusaj chunka*], 90 [*isqun chunka*], 100 [*pachak*], 1000 [*waranqa*], 10000 [*chunka waranqa* o *huno*], 10 × 10000 [*chunka huno*], 100 × 10000 [*pachak huno*], 1000 × 10000 [*waranqa huno*]. Los *quioucamayoc* o historiadores de los Incas según Cobo (1892), no podían desconocer los aspectos relacionados con el gobierno, los ritos y las costumbres de su propio pueblo, ya que habían vivido durante la época de los reyes Incas y estaban familiarizados con todo aquello sobre lo cual fueron evaluados. Esto se debía a los memoriales de sus quipus y pinturas que aún perduraban. Particularmente la que tenían en un templo del Sol, junto a la ciudad del Cuzco, "de la cual historia tengo para mí se debió de sacar una que yo vi en aquella ciudad dibujada en una tapicería de *cumbe*, no menos curiosa y bien pintada que si fuera de muy finos paños de corte" (Cobo, 1892, p. 117).

A.4 Quilca

Denominados también *quellca* o *quellcca*, mencionados en los primeros diccionarios de quechua como papel, o carta (Ricardo, 1586) y escritura (Gonzalez-Holguin, 1608). Consistían en símbolos pictográficos e ideográficos que se pintaban o tallaban en superficies como cerámica, piedras y textiles. Estos símbolos representaban ideas, objetos y conceptos, y se utilizaban para registrar eventos importantes, contar historias y transmitir conocimientos culturales y religiosos. Es menos documentado y comprendido en comparación con los quipus, pero forman parte del conjunto de métodos de comunicación y registro desarrollados por las culturas precolombinas en la región andina.

A.5 *Tocapu*

Los Incas, exactamente como los chinos y los egipcios, se sirvieron de formas geométricas que provienen de la naturaleza para fijar su sabiduría, basándose en datos etnográficos recogidos en la comunidad quechua hablante de *Q'ero* (Cuzco, Perú) y la túnica de Dumbarton Oaks, (Silverman, 2011) precisa que los *Q'eros* usan formas geométricas para

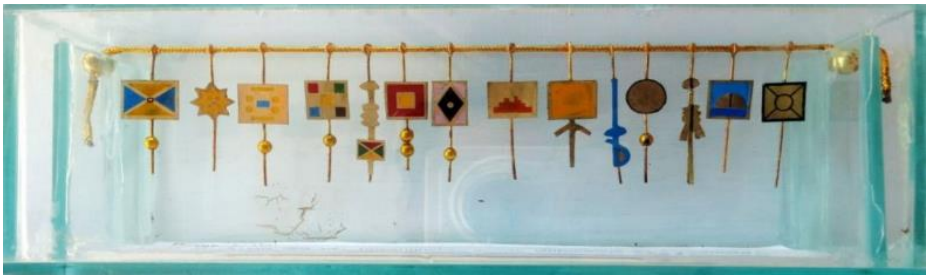
decorar sus textiles. Finalmente indica el nombre, significado y categoría gramatical de estos diseños Incas, especificando nombres en singular y plural, dos sufijos y un adjetivo.

B. Escritura Inca

El sistema de la escritura *tawantinsuyana* es la más antigua del continente americano basado en un conjunto de importantes símbolos (Jara, 1975). *Teqsi simi* es la escritura Inca, trasmisión escrita a base de figuras simbólicas, palabra por palabra de una oración completa en forma horizontal llamado cardinales horizontales (Huillca-Huallpa, 2016). Está basada en diversas formas, trazos, códigos, figuras y colores estéticos. Encontrándose por lo general ocho bases en los cuales estructuraron o formaron su escritura. Estas bases son: cuadrangular y o cuadrado, rectangular o rectángulo, rectángulo inverso, circulo – estrella, rombo, rombo inverso, línea circular media o cuarta, nudos humanos u otros seres estilizados, entre otros. La pronunciación (fonética) vive hasta hoy, el cual es hablado o pronunciado diariamente en el *Tawantinsuyo* (Huillca-Huallpa, 2016); aunque diferente a los conceptos que tenemos en la actualidad sobre escritura como es el alfabeto, los Incas tenían escritura una basada en pictografías (dibujos de objetos) e iconografías (imágenes artísticas), las que plasmaron en tejidos, cerámica, vestimentas, muros, chullpas y que aún se conservan en los sitios donde vivieron (Silverman, 2011).

Figura 2

Qhapaq khipu teqsi simi

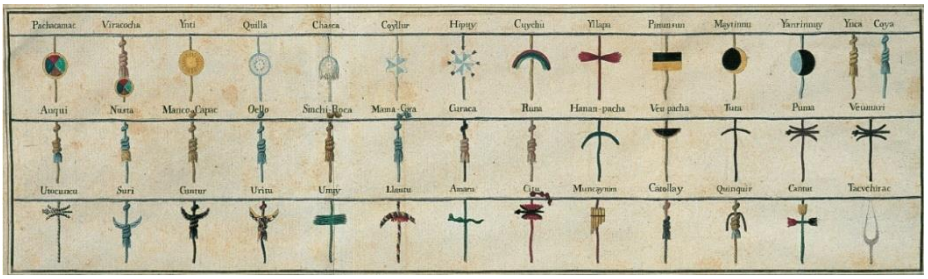


Nota. La ilustración representa el *qhapaq khipu teqsi simi* escrito con pequeñas láminas de oro. Tomado de *Qhapaq Khipu Teqsi Simi* (p. 59), por Huillca-Huallpa (2016).

Los antiguos peruanos posiblemente escribieron palabras y oraciones, usando un conjunto de palabras primarias (axiomas, en la jerga de *L-Systems*) para generar nuevas palabras, usando el conjunto específico de sílabas de las palabras primarias. Iconos u ornamentos especiales, como discos dorados, pequeños arcos pintados y otros diseños de joyería, insertados al principio de la cadena de nudos, identificaban cada palabra primaria y hacían posible la identificación del conjunto correcto de sílabas y finalmente, la correcta comprensión de la nueva palabra (Roncoroni, 2017). Podemos clasificarlos en dos grupos: aquellos que creen que la escritura en textiles precede al incanato y aquellos que consideran que es contemporánea a este. Las investigaciones en este ámbito se centran en los *tocapu*, que son motivos geométricos cuadrados o rectangulares que adornan los tejidos. Estos patrones también aparecen en vasos ceremoniales (keros), así como en la cerámica y la piedra. Otros trabajos en cambio muestran que todavía hoy en algunas comunidades peruanas, alejadas de los centros civilizados, se utiliza el tejido como medio para trasmitir mensajes y fijar acontecimientos (Tanodi, 2000). El quipu con inserciones de tipo *tocapu* fueron discutidas e ilustradas por primera vez en la obra *Lettera Apologetica* de *Raimondo di Sangro*, príncipe de Sansevero (véase Figura 3).

Figura 3

Ilustración original del quipu con inserciones de tipo tocapu



Nota. Ilustración original de *Lettera apologetica* de *di Sangro*, que muestra palabras principales y sus íconos. Tomado de *Quipus, computation and generative grammars* (p. 359), por Roncoroni (2017), GA2018 – XXI Generative Art Conference.

B.1 Escritura simbólica

Representación pictórica, ideográfica, geométrica, artística y abstracta; única y original, con pronunciación y simbolismo propio, por ello algunas palabras simbólicas no se pueden traducir al idioma y escritura hispana en su integridad (Huillca-Huallpa, 2016), su importancia se basa en:

- Cada símbolo de su escritura significa una vocal o una sílaba o una palabra o varias palabras.
- Es poli semántica, de varios significados de acuerdo a la forma de expresión u oración.
- Una misma figura simbólica puede tener diferente significado de acuerdo a la posición donde se encuentre en el espacio geométrico o figura geométrica.
- La fonología y grafías varían en cada asentamiento cultural o civilización en el mundo.
- Un solo símbolo puede decir todo un mensaje.
- No se requiere de artículos, preposiciones, etc., porque su fonología fue frondosa en sonidos que se interrelacionaban en una capacidad de síntesis en el mensaje.

B.2 Escritura pictográfica

Los Incas usaban la escritura pictográfica, que proviene de formas de la naturaleza u objetos concretos hechos por el hombre, para fijar su sabiduría antes de la llegada de los españoles (Silverman, 2011).

B.3 Escritura ideográfica

Se fundamenta en el uso de símbolos que representan un concepto o una idea, en lugar de un sonido, este tipo de escritura se reduce mucho ya que en un solo signo podemos encontrar una frase extensa o una idea amplia. Por los antecedentes prehispánicos ofrecidos y por la conexión con la vida agrícola y familiar, es muy probable que la escritura ideográfica tenga origen prehispánico e incluso preincaico (Garcés y Sánchez, 2014).

B.4 Escritura geométrica

Los Incas, exactamente como los chinos y los egipcios, se sirvieron de formas geométricas que provienen de la naturaleza para fijar su sabiduría (Silverman, 2011). En las grafías geométricas encontramos los elementos lineales básicos de una plástica abstracta, que al mismo tiempo son escritura, una síntesis de palabra e imagen (Boido, 2012).

B.5 Escritura artística y abstracta

La escritura artística es la terapia esencial de ser para los otros (Peñuela, 2017). Así esta escritura artística es el resultado de una doble función, primero la de dejar asentado algo, y segunda la de denotar artísticamente dicho conglomerado de hechos convertidos ahora en signos, para mostrar en su pictografía la relación escritura-pintura que llamaremos glifo-figura (Caso, 1989). La escritura abstracta y/o dibujo escrito son obras que imprimen la sonoridad de la lengua en objetos, lienzos o papel (Rodríguez, 2015), que permitía a las obras de esta índole identificarse con la naturaleza (Ochoa y Castrillón, 2006).

B.6 Escritura logosilábica

Sistemas cuyos signos representan palabras enteras o sílabas individuales (Hyland y Hyland, 2020).

B.7 Escritura cifrada

Es la escritura en donde el texto no tiene un significado evidente, ocultando el verdadero significado de los signos escritos. Puede afectar a letras, grupos de letras, sílabas, palabras, frases, etc. o hacerse de manera híbrida (González, 2014).

1.1.2 Redes neuronales convolucionales profundas

A. Procesamiento de imágenes

El procesamiento de imágenes es el conjunto de técnicas, algoritmos y operaciones matemáticas que se utilizan para mejorar la calidad, extraer información, manipular y transformar imágenes digitales.

A los seres humanos les gusta que sus imágenes sean nítidas, claras y detalladas; las máquinas prefieren que sus imágenes sean sencillas y depuradas (McAndrew, 2016).

A.1 Imagen digital

Una imagen es la representación visual de un objeto de la vida real (una persona o cualquier otro objeto) en forma bidimensional (Singh, 2019). Los píxeles son los componentes básicos de una imagen digital (Demaagd et al., 2012). Un píxel es considerado como el color o la intensidad de la luz (Rosebrock, 2017). Una imagen digital es una matriz bidimensional de píxeles, cada píxel consta de un número de bits (Furht et al., 2018), está formada por $M \times N$ píxeles y cada uno de estos píxeles está representado por k bits (Tyagi, 2018). El número de píxeles de una imagen define la resolución y determina la calidad de la imagen.

A.2 Tipos de imágenes

Las imágenes en escala de grises son imágenes monocromáticas que se compone de píxeles en diferentes tonos de gris, desde el negro hasta el blanco (Q. Zhang y Skjetne, 2018). Las imágenes a color son imágenes en formato *RGB* (rojo, verde, azul) (Rafael C y Richard E, 2018). Las imágenes binarias son imágenes con solo dos colores (blanco y negro).

A.3 Transformaciones básicas

La transformación de espacio de color implica la conversión entre distintos espacios de color (Q. Zhang y Skjetne, 2018; Thanki y Kothari, 2019; Iftekharuddin y Awaal, 2012).

La transformación geométrica comprende la rotación (Chung, 2017), selección (Kinser, 2018), traslación, redimensión y enmascaramiento (González-Acuña et al., 2021; Ranjan y Senthamilarasu, 2020), recortar, invertir y enmarcar (Parkin, 2018), escalado (Tyagi, 2018), interpolación (Li, 2018), seccionar (Demaagd et al., 2012), corte y transponer (Distante y Distante, 2020b), deformar (Solomon y Breckon, 2011).

A.4 Ruido en las imágenes

Las imágenes suelen verse afectadas por el ruido, que es cualquier elemento que degrade la imagen ideal. Este ruido puede tener un impacto significativo en el procesamiento de las imágenes. El ruido es causado por el entorno, el dispositivo de imagen, la interferencia eléctrica, el proceso de digitalización, y otros. Para una imagen $f(i,j)$ la relación señal/ruido se define como sigue (Dawson-Howe, 2014):

$$S/N_{ratio} = \sum_{(i,j)} f^2(i,j) \Big/ \sum_{(i,j)} v^2(i,j)$$

donde $v(i,j)$ es el ruido. Los tipos de ruido más comunes son el ruido espacial que comprende el ruido Gaussiano y ruido uniforme (Dawson-Howe, 2014; Thanki y Kothari, 2019; Y.-J. Zhang, 2022), ruido impulsional o "sal y pimienta" (Dawson-Howe, 2014), ruido Rayleigh, ruido Erlang/Gamma, ruido exponencial, ruido Poisson (Makitalo y Foi, 2010), ruido Lognormal, ruido Spekle, grano o estructural (Garcia Chilan y Viteri Paredes, 2010); ruido periódico (Aizenberg y Butakoff, 2008; Xie et al., 2012; Hamd, 2014), ruido de cámara y ruido de colores (Kinser, 2018).

A.5 Histograma de imagen

Es una abstracción de una imagen donde se determina la frecuencia de cada valor de imagen (contraste/brillo/intensidad) (Burger y Burge, 2010; Dawson-Howe, 2014; Demaagd et al., 2012; Rosebrock, 2016). Un histograma representa la distribución de intensidades de píxeles (ya sea en escala de grises o en color) en una imagen. Se puede visualizar como un gráfico (o diagrama) que brinda una intuición de alto nivel de la distribución de intensidad (valor de píxel) (Rosebrock, 2016).

A.6 Técnicas de procesamiento de imágenes

El filtrado de imágenes es una técnica utilizada con el fin de modificar, mejorar, corregir o alterar su apariencia, algunos de los filtros más utilizados son los filtros de suavizado como el filtro de media, filtro

de la mediana, filtro máximo, filtro mínimo (Chityala y Pudipeddi, 2020), filtro de orden o rango, filtro Gaussiano (Solomon y Breckon, 2011), filtro bilateral (Mordvintsev y Abid, 2014; Villán, 2019), filtro Gabor (Mora Lazarini, 2008); además están los filtros de la primera derivada como el filtro Roberts (Sánchez, 2010), filtro Sobel (Chityala y Pudipeddi, 2020), filtro Prewitt, filtro Kirsch, filtro Robinson, Frei-Chen (Isotrópico), Marr-Hildreth (Shrivakshan y Chandrasekar, 2012), filtro Canny (Rebaza, 2007; Sánchez, 2010), filtro Laplaciano, filtro de Laplaciano de una Gaussiana (Chityala y Pudipeddi, 2020), filtro Zero-crossing (Solomon y Breckon, 2011); los filtros de detección de formas como el filtro Frangi (Frangi et al., 1998). La morfología matemática se aplica principalmente a imágenes binarias mediante operaciones como dilatación y erosión, aunque puede utilizarse en todo tipo de imágenes (Solomon y Breckon, 2011; Thanki y Kothari, 2019). La segmentación consiste en identificar y extraer estructuras subyacentes para facilitar su interpretación, como bordes o agrupaciones de píxeles que forman regiones con propiedades específicas (Cervantes y Miguel, 2019). La segmentación es el nombre dado al proceso genérico por el cual una imagen se subdivide en sus regiones o categorías que corresponden a diferentes objetos o partes de objetos (Dey, 2018; Solomon y Breckon, 2011), en función de algunos atributos de la imagen (Thanki y Kothari, 2019). El proceso de dividir la imagen en regiones homogéneas, donde se agrupan todos los píxeles que corresponden a un objeto, se denomina segmentación (Distante y Distante, 2020). La mayoría de los algoritmos de segmentación se basan en una de las dos propiedades básicas de los valores de intensidad de imagen: discontinuidad y similitud (Rafael y Richard, 2018). Se aborda a través de métodos de borde/límite o métodos basados en histogramas o métodos basados en regiones. En la segmentación son utilizados filtros de umbralización como el filtro de Otsu (Chityala y Pudipeddi, 2020).

A.7 Conjunto de datos de imagen

Un conjunto de datos de imagen es una recopilación de representaciones pictóricas pertenecientes a una o más clases,

almacenados en memoria o en la nube; y están configurados para ser entrenados en un modelo.

B. Visión por computadora

La visión por computadora es una rama de la inteligencia artificial enfocada en entrenar computadoras para ver y procesar imágenes y videos digitales de la misma manera que la visión humana (Vaughan, 2023), ayuda a las computadoras a ver y comprender el significado de las imágenes digitales, como fotografías y videos (Brownlee, 2019), incluye la funcionalidad para adquirir, procesar y analizar imágenes digitales (Villán, 2019); determina lo que la computadora está viendo en el entorno circundante (Ranjan y Senthamilarasu, 2020).

Es el proceso de aplicar una gama de tecnologías y métodos para proporcionar inspección automática basada en imágenes, control de procesos y guía robótica en aplicaciones industriales (González-Acuña et al., 2021), aborda el lado visual del problema del aprendizaje automático (Kulkarni et al., 2022), y es un componente clave en los robots (Sundararajan, 2017). La visión por computadora no solo es aplicable en automóviles autónomos, sino también en reconocimiento facial, detección de objetos, reconocimiento de escritura a mano, detección de números de placas, imágenes médicas, reconstrucción de imágenes, síntesis de imágenes, transferencia de estilo de imagen y segmentación de objetos, entre otras cosas (Ranjan y Senthamilarasu, 2020). Aunque la gente había ideado muchas arquitecturas de red diferentes para resolver la visión por computadora, las redes neuronales convolucionales las superaron a todas (Thomas y Passi, 2019).

B.1 Red neuronal convolucional (*CNN*)

Una *CNN* es un algoritmo de aprendizaje profundo que puede tomar una imagen de entrada, asignar importancia (pesos y sesgos aprendibles) a varios aspectos/objetos en la imagen, y poder diferenciar uno del otro (El-Amir y Hamdy, 2020), tienen la capacidad de aprender filtros/características. El preprocesamiento requerido en una *CNN* es

significativamente menor en comparación con otros algoritmos de clasificación. Su arquitectura se basa en realizar de manera secuencial una operación denominada convolución (Sotelo, 2021). Adicional a la convolución se realizan otras operaciones como la aplicación de una función de activación como *ReLU* y una etapa de reducción de dimensiones como el *maxpooling.* La red neuronal de convolución es un modelo matemático del córtex visual biológicamente inspirado. Comenzó con éxito con el trabajo de David Hubel y Torsten Wiesel, quienes ganaron el Premio Nobel de Fisiología o Medicina de 1981 por este trabajo (Chityala y Pudipeddi, 2020). El análisis de la corteza visual del mensaje codificado de la retina procede como si ciertas celdas leyeran las letras simples en el mensaje y las compilen en sílabas que posteriormente son leídos por otras celdas, que, a su vez, compilan las sílabas en palabras, y estas son finalmente, leídas por otras células que compilan palabras en oraciones que se envían a los centros superiores del cerebro, donde se origina la impresión visual y se almacena la memoria de la imagen (Karolinska, 1981).

La comprensión de la corteza visual del cerebro allanó el camino para el modelado matemático de la vía visual. El primer trabajo exitoso fue realizado por Kunihiko Fukushima (1980). Demostró un modelo jerárquico utilizando convolución y muestreo descendente. La convolución permitió la visualización de solo una parte de la imagen o el video durante el procesamiento. El muestreo descendente se realizó promediando. Muchos años más tarde, se introdujo un método diferente llamado *maxpooling* que todavía se usa hoy. El siguiente gran avance fue el trabajo de Yann LeCun et al. (1989) quien introdujo un enfoque de *backpropagation* para aprender los parámetros de una *CNN*. Las redes neuronales convolucionales son usadas en tareas de visión por computador. Una red neuronal convolucional profunda consta de muchas capas. Normalmente se alternan dos tipos diferentes de capas, convolucional y de agrupación. La profundidad de cada filtro aumenta de izquierda a derecha en la red. La última etapa suele estar formada por una o más capas completamente conectadas (Gulli y Pal, 2017).

B.1.1 Componentes en la arquitectura de una *CNN*

En una arquitectura *CNN*, hay varias capas convolucionales y capas de agrupación al principio, que se utilizan principalmente para simplificar la complejidad de los datos de la imagen y reducir sus tamaños. Además, son muy útiles para extraer patrones complejos de los patrones básicos observados en las imágenes. Después de usar varias capas convolucionales y de agrupación (compatibles con funciones de activación), remodelamos nuestros datos de matrices bidimensionales o tridimensionales en una matriz unidimensional con una capa aplanada. Después de la capa aplanada, un conjunto de capas completamente conectadas toman la matriz unidimensional aplanada como entrada y se completa la tarea de clasificación o regresión (Yalçın y Yalçın, 2021).

a. Imagen de entrada

Una imagen de entrada forma el primer componente de una arquitectura *CNN*. Una imagen puede ser de cualquier tipo: un ser humano, un animal, un paisaje, una imagen médica de rayos X, entre otros (Moocarme et al., 2020).

b. Capas de convolución

Una convolución es básicamente una multiplicación entre dos matrices (Ayyadevara y Reddy, 2020). La matriz de la imagen de entrada es multiplicada con el detector de características o filtro, en todas las áreas de los datos de entrada para tener como resultado el mapa de características. Matemáticamente la ecuación de salida de la convolución está dada por (Sotelo, 2021):

$$Conv_{out} = \sum_{i=1}^{F*F} p_i w_i + b$$

donde F es el tamaño del filtro, siendo $F * F$ el total de componentes, p_i son los píxeles de la porción de la imagen a procesar, w_i son los parámetros del filtro y b es peso de tendencia o sesgo (*bias*).

b.1 Detector de características o filtro

Los filtros se pueden considerar como versiones más pequeñas de la imagen que se componen de valores de peso que se pueden aprender (Liu y Mehta, 2019). El detector de características o filtro es una matriz o patrón que colocas en una imagen para transformarla en un mapa de características (Moocarme et al., 2020). Este detector de características se coloca (superpone) sobre la imagen original y el cálculo se realiza multiplicando los elementos correspondientes.

b.2 Mapa de características

Es la imagen reducida que se produce por la convolución de una imagen y un detector de características. Tenemos que poner el detector de características en todas las ubicaciones posibles de la imagen original y derivar una imagen más pequeña de ella; esa imagen derivada es el mapa de características de la imagen de entrada (Moocarme et al., 2020). El detector de características es el filtro y el mapa de características es la imagen reducida, parte de la información se pierde al reducir la imagen. En una *CNN* real, se utilizan varios detectores de características para producir una serie de mapas de características. Los filtros en una capa convolucional son ajustados por el modelo durante el entrenamiento.

b.3 Profundidad del filtro

La profundidad del filtro generalmente se refiere al número de canales en la imagen (Rosebrock, 2017). Para los filtros de las capas posteriores, la profundidad corresponde al número de filtros aplicados en las capas anteriores (Ketkar y Moolayil, 2021).

b.4 Número de filtros

Los filtros actúan como un extractor de características; por lo tanto, es común tener varios filtros dentro de cada bloque convolucional de la red. Una disposición de muestra sería un bloque convolucional con 32 filtros de tamaño 3×3 (y de profundidad 3) seguido de activación/normalización por lotes y bloques de agrupación, seguido de otro bloque con 64 filtros (ahora con una profundidad de 32), así sucesivamente (Ketkar y Moolayil, 2021).

b.5 Tamaño de filtros

En general, para operaciones de convolución, se usan filtros de tamaño 3×3 y 5×5. Las implementaciones anteriores también favorecían los filtros 7×7 (Ketkar y Moolayil, 2021).

b.6 Filtrado

El filtrado se realiza multiplicando cada valor de una parte de los datos de la imagen por el valor de filtro correspondiente (Yalçın y Yalçın, 2021).

b.7 Paso

Hasta ahora, asumimos que el deslizamiento del filtro ocurre un píxel a la vez, pero no siempre es así. Podemos deslizar el filtro varias posiciones. Este parámetro de las capas convolucionales se llama paso (Vasilev et al., 2019). Paso es el número de posiciones sobre las que deslizamos el filtro en el segmento de entrada en cada paso. De forma predeterminada, el paso es 1. Si es mayor que 1, lo llamamos convolución de paso. El paso más grande aumenta el campo receptivo de las neuronas de salida. Con el paso 2, el tamaño del segmento de salida será aproximadamente cuatro veces más pequeño que el de entrada (Vasilev, 2019). El uso de un valor paso grande disminuiría el número de cálculos de filtro. Un valor paso grande reduciría significativamente la complejidad del modelo, pero podríamos perder algunos de los patrones a lo largo del proceso. Por lo tanto, siempre debemos establecer un valor paso óptimo, ni demasiado grande ni demasiado pequeño (Yalçın y Yalçın, 2021). Para generar la imagen de salida, el filtro se desplaza sobre la imagen para procesar otros píxeles de la imagen de entrada y, de esta manera, ir generando la imagen de salida. El tamaño del desplazamiento lo simbolizamos como S (paso o *stride*). Si I es el tamaño de la imagen de entrada, F el tamaño del filtro, el tamaño de la imagen de salida O lo calcularemos con la ecuación (Sotelo, 2021):

$$O = \frac{I - F}{S} + 1$$

Al aplicar la convolución sobre una imagen, la salida es de menor tamaño, y esto es contraproducente si deseamos hacer una red neuronal profunda, pues lo ideal es conservar el tamaño para poder realizar la mayor cantidad de etapas de extracción de características. Para lograr que el tamaño de la imagen de salida sea igual al de entrada, rellenamos la imagen de entrada alrededor con ceros (0). Esto es denominado *zero-padding* (Sotelo, 2021).

b.8 *Zero-Padding*

El *zero-padding* permite conservar el tamaño de la imagen sin importar la aplicación de la convolución (Sotelo, 2021). Esto se logra rellenando los bordes con ceros para conservar el tamaño original de la imagen al aplicar una convolución; lo mismo se aplica a los filtros dentro de una *CNN* (Rosebrock, 2017). Agregar ceros alrededor del borde es equivalente a agregar un borde negro alrededor de una imagen (Ranjan y Senthamilarasu, 2020). También podemos establecer el relleno en 2 si es necesario. Podemos calcular *zero-padding* P, usando la siguiente expresión (Sotelo, 2021):

$$P = \frac{F - 1}{2}$$

Considerando estos ceros de rellenos, se tiene la fórmula que nos permite calcular el tamaño de la imagen de salida.

$$O = \frac{I - F + 2P}{S} + 1$$

b.9 Convolución 1D

La mayoría de las convoluciones simplistas son convolucionales 1D que generalmente se usan en conjuntos de datos de secuencia. Se pueden usar para extraer subsecuencias 1D locales de las secuencias de entrada e identificar patrones locales dentro de la ventana de convolución. Otros usos comunes de las convoluciones 1D se ven en el área de *NLP*, donde cada oración se representa como una secuencia de palabras. La convolución 1D está dada por la siguiente ecuación.

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n-k]$$

b.10 Convolución 2D

La idea principal de las convoluciones 2D es que el filtro convolucional se mueve en 2 direcciones (x, y) para calcular características de baja dimensión a partir de los datos de la imagen. La forma de salida es también una matriz bidimensional. La convolución 2D está dada por la siguiente ecuación.

$$y[m,n] = x[m,n] * h[m,n]$$

$$y[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \cdot h[m-i, n-j]$$

b.11 Convolución 3D

Las convoluciones 3D aplican un filtro tridimensional al conjunto de datos y el filtro se mueve en 3 direcciones (x, y, z) para calcular las representaciones de características de bajo nivel. Su forma de salida es un espacio de volumen tridimensional. Son útiles en la detección de eventos en videos, imágenes médicas en 3D, entre otros. No se limitan al espacio 3D, sino que también se pueden aplicar a entradas de espacio 2D, como imágenes. La convolución 3D está dada por la siguiente ecuación.

$$y[m,n,o] = x[m,n,o] * h[m,n,o]$$

$$y[m,n,o] = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j,k] \cdot h[m-i, n-j, o-k]$$

b.12 Convoluciones dilatadas

Las convoluciones dilatadas o altrosas definen el espacio entre los valores en un núcleo. En este tipo de convolución, la vista receptiva de los núcleos aumenta debido al espaciado. La complejidad sigue siendo la misma, pero en este caso se generan características diferentes. La siguiente fórmula explica la convolución dilatada $*_l$ (Yu y Koltun, 2015):

$$s(t) = (f *_l g)(t) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(t - l\tau)$$

donde l es un valor entero positivo llamado factor de dilatación. Las convoluciones dilatadas pueden aumentar exponencialmente el tamaño del campo receptivo sin perder resolución o cobertura (Vasilev, 2019).

b.13 Convolución con imágenes a color

Hasta el momento se ha visto la convolución con imágenes de un canal o de escala de grises. Si trabajamos con imágenes a color, normalmente se representan con tres canales correspondientes a los colores rojo, verde y azul (Sotelo, 2021). El filtro ya no tiene una sola matriz, sino tres para poder interactuar con los tres canales; cada matriz del filtro hace una operación similar al caso de escala de grises y todos los valores se suman para finalmente generar un valor en la imagen de salida resultante.

c. Capas de activación

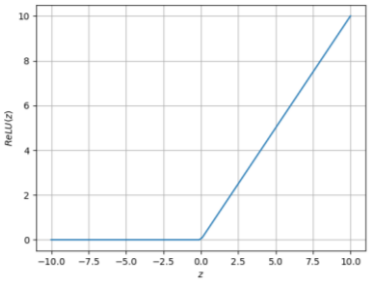
Después de cada capa de convolución en una *CNN*, aplicamos una función de activación no lineal, como *ReLU*, *ELU* o cualquiera de las otras variantes (Rosebrock, 2017).

c.1 *ReLU*

La función de activación más usada en las *CNN* es *ReLU* (Sotelo, 2021). Los valores negativos de entrada se rectifican a cero y los valores positivos pasan sin modificaciones a la salida, véase la Figura 4.

Figura 4

Función de activación ReLU



Nota. Activación *ReLU* (función de activación más utilizada para redes neuronales profundas). Adaptado de *Deep learning for computer vision with python: Starter bundle* (p. 125), por Rosebrock (2017), PyImageSearch.

d. Normalización por lotes

La normalización por lotes (*batch normalization*) se utiliza para normalizar las activaciones de un volumen dado de entrada antes de pasarlo a la siguiente capa de la red (Rosebrock, 2017). Ayuda a que nuestras redes funcionen mejor en general y aprendan más rápido. Al usar la normalización por lotes, para cada minilote, podemos normalizar ese lote o *batch* para que tenga una media de 0 y una varianza unitaria, después (o antes) de cada no linealidad.

Esto permite que cada capa tenga una entrada normalizada para aprender, lo que hace que esa capa sea más eficiente en el aprendizaje (Bernico, 2018). En la normalización por lotes se calcula la media μ_B y la desviación estándar σ_B^2 de la información que fluye por la capa.

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} f_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(f_i - \mu_B)^2$$

donde m es el tamaño del minilote, f_i es la activación del filtro convolucional ante *i-ésimo* patrón del minilote.

La normalización estadística está dada por (Sotelo, 2021):

$$f_{iNor} = \frac{f_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

donde f_{iNor} es la activación del filtro convolucional ante *i-ésimo* patrón del minilote normalizado, ε es la constante usada para estabilidad numérica (evitar división por cero). Esta podría ser la salida de la capa ya normalizada; sin embargo, para ajustarse a las condiciones de los datos, se utiliza una operación adicional de escalado y desplazamiento usando dos parámetros, denominados φ y η.

Estos parámetros son entrenables y se modifican de manera similar a como se entrenan las conexiones sinápticas y umbrales de las diferentes neuronas. La siguiente ecuación aplica esta operación, y este valor constituye la salida de la capa ya normalizada (Sotelo, 2021).

$$f_{iBN} = \varphi f_{iNor} + \eta$$

e. Capas de agrupación o reducción

La capa de agrupación ayuda a reducir el tamaño de los datos para facilitar el cálculo (Alla y Adari, 2019); se utilizan para reducir la dimensionalidad de una red convolucional a medida que se agregan capas de convoluciones (Bernico, 2018), lo que reduce el sobreajuste (Shanmugamani, 2018). Tienen el beneficio adicional de hacer que los detectores de características sean algo más robustos. Las capas de agrupación dividen una matriz en secciones que no se superponen y, luego, normalmente toman el valor máximo (agrupación máxima) en cada región. Alternativamente, se puede emplear el valor promedio (agrupación promedio) en cada región; sin embargo, rara vez se usa.

e.1 Agrupación máxima

La agrupación máxima propaga el valor máximo de los valores de entrada del campo receptivo (Vasilev, 2019).

e.2 Agrupación promedio

La agrupación promedio propaga el valor promedio de las entradas en el campo receptivo (Vasilev, 2019).

e.3 Agrupación promedio global

La agrupación promedio global es lo mismo que la agrupación promedio, pero la región de agrupación tiene el mismo tamaño que el mapa de características. La agrupación promedio global realiza un tipo extremo de reducción de dimensionalidad, la salida es un único escalar, que representa el valor promedio de todo el mapa de características (Vasilev, 2019).

f. Capa clasificadora

El resultado de la última capa convolucional son las características de alto nivel que representan el patrón de entrada. Estas características se pasan por una o varias capas totalmente conectadas para generar la clasificación que necesitamos a la salida de la *CNN* (Sotelo, 2021).

f.1 Aplanado

Aplanado es una capa en la que toda la entrada se aplasta en una sola dimensión (Alla y Adari, 2019).

f.2 Capa totalmente conectada

La primera capa totalmente conectada recibe como entrada la salida de la última capa convolucional; estas se procesan de la manera habitual (multiplicación de pesos por las características entregadas por la última capa convolucional). Este proceso se repite, dependiendo de la cantidad de capas totalmente conectadas que tengamos. Las activaciones de la última capa se pasan por una capa de neuronas con funciones de activación *softmax* (Sotelo, 2021). Es común utilizar una o dos capas completamente conectada antes de aplicar el clasificador *softmax* (Rosebrock, 2017).

f.3 *Dropout*

Los métodos de regularización están diseñados para disminuir el sobreajuste (*overfitting*) de los modelos. Un modelo sobreajustado memoriza los datos de entrenamiento, pero no puede predecir correctamente nuevas observaciones. Entre los numerosos métodos disponibles, se destacan la regularización L1 y L2 (*weight decay*) y el *dropout*. El *dropout* es una técnica de regularización en la que una

proporción de nodos seleccionados al azar se eliminan o se ignoran durante el proceso de entrenamiento (Alla y Adari, 2019); tiene como objetivo ayudar a prevenir el sobreajuste aumentando la precisión de las pruebas, quizás a expensas de la precisión del entrenamiento (Rosebrock, 2017). Es más común colocar capas *dropout* con probabilidad $p = 0.5$ entre capas completamente conectadas de una arquitectura donde se supone que la capa completamente conectada final es nuestro clasificador *softmax*, también podemos aplicar la probabilidades de abandono más pequeñas (ejemplo, $p = 0.10 - 0.25$) también en capas anteriores de la red (normalmente después de una operación de reducción de muestreo, ya sea mediante agrupación máxima o convolución) (Rosebrock, 2017).

f.4 *Softmax*

La función *softmax* calcula las probabilidades de salida final para cada clase (Rosebrock, 2017). Cuando la red está entrenada, la clase se asigna a la salida que tenga la mayor probabilidad. P_k es la probabilidad de la *k-ésima* clase. En el proceso de entrenamiento, se modifican los pesos para garantizar que la probabilidad sea la mayor en la clase deseada (Sotelo, 2021). Para la modificación de los pesos de la *CNN*, se usa la función de pérdida *Categorical Cross Entropy Loss* (Chauhan et al., 2018; Ho y Wookey, 2019).

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \cdot \log\left(f(x_{ij})\right)$$

B.1.1 Arquitecturas *DCNN*

Las arquitecturas de redes neuronales convolucionales están diseñadas para procesar datos con estructura matriciales, como las imágenes. Estas redes están compuestas por capas convolucionales que aplican filtros para extraer características locales, capas de agrupamiento que reducen la dimensionalidad, y capas totalmente conectadas que se encargan de la clasificación final. Su capacidad para aprender representaciones efectivas ha permitido su uso en diversas aplicaciones de visión por computadora, incluyendo la clasificación de imágenes y la

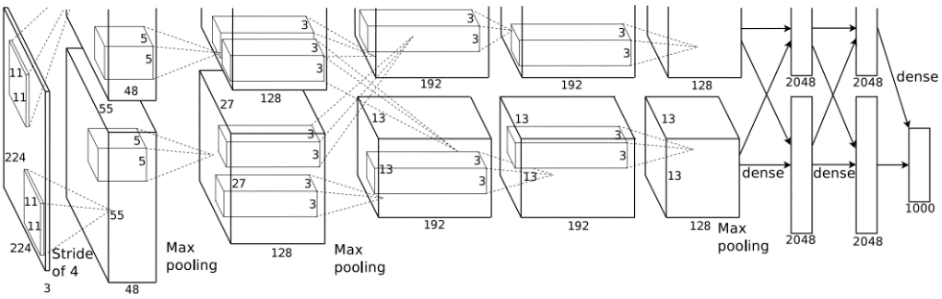
detección de objetos (Goodfellow et al., 2016; Krizhevsky et al., 2017; LeCun et al., 2015)

a. ***AlexNet***

AlexNet es una red neuronal convolucional para la clasificación de imágenes. Obtuvo una buena puntuación en el desafío de *ImageNet* en 2012 (Ekman, 2021). La red consta de ocho capas con pesos; las primeras cinco son capas convolucionales, mientras que las tres últimas son capas completamente conectadas.

Figura 5

Arquitectura CNN AlexNet



Nota. Distribución de tareas entre las dos *GPU*. Una *GPU* procesa las secciones de la capa en la parte superior de la figura, mientras que la otra maneja las secciones de la capa en la parte inferior. Las *GPU* se comunican solo en ciertas capas. La entrada de la red tiene 150528 dimensiones, y el número de neuronas en las capas restantes de la red es de 253440, 186624, 64896, 64896, 43264, 4096, 4096 y 1000 neuronas. Tomado de *ImageNet classification with deep convolutional neural networks* (p. 5), por Krizhevsky et al. (2017b).

La salida de la última capa completamente conectada se dirige a un *softmax* con 1000 unidades, que produce una distribución sobre las 1000 etiquetas de clase. La red optimiza el objetivo de la regresión logística multinomial, lo que equivale a maximizar el promedio de la probabilidad logarítmica de la etiqueta correcta durante el entrenamiento, de acuerdo con la distribución de predicción. Los núcleos de las capas convolucionales segunda, cuarta y quinta se conectan Solo se conectan a los mapas de características de la capa anterior que están en la misma *GPU*. Los núcleos de la tercera capa convolucional están conectados a todos los mapas de características de la segunda capa. Las neuronas en las

capas completamente conectadas se enlazan con todas las neuronas de la capa previa. Las capas de normalización de respuesta siguen a las primeras dos capas convolucionales, mientras que las capas de agrupamiento máximo siguen tanto a las capas de normalización de respuesta como a la quinta capa convolucional. La función de activación no lineal *ReLU* se aplica a la salida de cada capa convolucional y de cada capa completamente conectada.

La primera capa convolucional procesa la imagen de entrada de 224×224×3 utilizando 96 núcleos de tamaño 11×11×3 con un paso de 4 píxeles (la distancia entre los centros de los campos receptivos de las neuronas vecinas en un mapa de características).

La segunda capa convolucional toma como entrada la salida (normalizada y agrupada) de la primera capa y la filtra con 256 núcleos de tamaño 5×5×48. Las capas convolucionales tercera, cuarta y quinta están conectadas directamente entre sí sin capas intermedias de agrupamiento o normalización.

La tercera capa convolucional está compuesta por 384 núcleos de tamaño 3×3×256, los cuales se conectan a las salidas (normalizadas y agrupadas) de la segunda capa convolucional. La cuarta capa convolucional dispone de 384 núcleos de tamaño 3×3×192, mientras que la quinta capa convolucional tiene 256 núcleos de tamaño 3×3×192. Las capas completamente conectadas tienen 4096 neuronas cada una (Krizhevsky et al., 2017).

b. *VGG11-BN*

La entrada es una imagen *RGB* de $224 \times 224 \times 3$ de tamaño fijo. La imagen se procesa a través de una serie de capas convolucionales que emplean filtros con un campo receptivo muy pequeño de 3×3, el tamaño mínimo necesario para capturar la información de izquierda/derecha, arriba/abajo y centro.

En una de las configuraciones, también se utilizan filtros de convolución 1×1, que se pueden interpretar como una transformación

lineal de los canales de entrada, seguida de una función no lineal. El paso de convolución se mantiene en 1 píxel, y el relleno espacial en la capa de convolución se ajusta para preservar la resolución espacial después de la convolución; por ejemplo, el relleno es de 1 píxel para capas de convolución de 3×3. La agrupación espacial se lleva a cabo mediante cinco capas de agrupamiento máximo que siguen a algunas de las capas convolucionales (no todas las capas convolucionales están seguidas de agrupamiento máximo).

La agrupación máxima se realiza con una ventana de 2×2 píxeles y un paso de 2. Una pila de capas convolucionales (que tiene una profundidad diferente en diferentes arquitecturas) es seguida por tres capas totalmente conectadas (*FC*): las dos primeras tienen 4096 canales cada una, la tercera realiza una clasificación *ILSVRC* de 1000 vías y, por lo tanto, contiene 1000 canales (uno para cada clase); la capa final es la capa *softmax* (Simonyan y Zisserman, 2015).

Figura 6

Configuraciones ConvNet

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Nota. Configuraciones de *ConvNet* (mostradas en columnas). La profundidad de las configuraciones aumenta de izquierda (A) a derecha (E) con la adición progresiva de más capas (en negrita). Los parámetros de la capa convolucional se detallan como "conv ⟨tamaño del campo receptivo⟩ - ⟨número de canales⟩". Tomado de *Very deep convolutional networks for large-scale image recognition* (p. 3), por Simonyan y Zisserman (2015), arXiv.

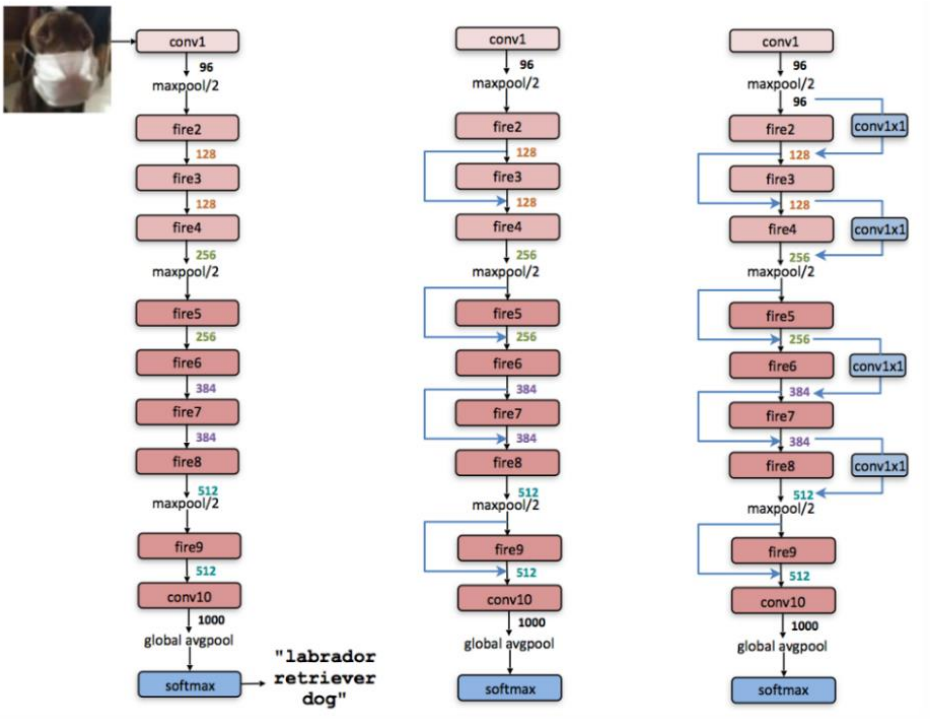
c. *SqueezeNet*

SqueezeNet inicia con una capa de convolución inicial (conv1), seguida de 8 módulos Fire (fire2-9), y finalizando con una capa de convolución terminal (conv10).

El número de filtros en cada módulo Fire aumenta progresivamente desde el comienzo hasta el final de la red. *SqueezeNet* aplica agrupamiento máximo con un paso de 2 después de las capas conv1, fire4, fire8 y conv10, esto se ilustra en la Figura 7.

Figura 7

Vista macroarquitectónica de SqueezeNet



Nota. Izquierda: *SqueezeNet*; Medio: *SqueezeNet* con derivación simple; Derecha: *SqueezeNet* con *bypass* complejo. Tomado de *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size* (p. 5), por Iandola et al. (2016), arXiv.

La arquitectura completa de *SqueezeNet* se detalla en la Figura 8.

Figura 8

Arquitectura completa de SqueezeNet

layer name/type	output size	filter size / stride (if not a fire layer)	depth	s_{1x1} (#1x1 squeeze)	e_{1x1} (#1x1 expand)	e_{3x3} (#3x3 expand)	s_{1x1} sparsity	e_{1x1} sparsity	e_{3x3} sparsity	# bits	#parameter before pruning	#parameter after pruning
input image	224x224x3										-	-
conv1	111x111x96	7x7/2 (x96)	1				100% (7x7)			6bit	14,208	14,208
maxpool1	55x55x96	3x3/2	0									
fire2	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	11,920	5,746
fire3	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	12,432	6,258
fire4	55x55x256		2	32	128	128	100%	100%	**33%**	6bit	45,344	20,646
maxpool4	27x27x256	3x3/2	0									
fire5	27x27x256		2	32	128	128	100%	100%	**33%**	6bit	49,440	24,742
fire6	27x27x384		2	48	192	192	100%	**50%**	**33%**	6bit	104,880	44,700
fire7	27x27x384		2	48	192	192	**50%**	100%	**33%**	6bit	111,024	46,236
fire8	27x27x512		2	64	256	256	100%	**50%**	**33%**	6bit	188,992	77,581
maxpool8	13x12x512	3x3/2	0									
fire9	13x13x512		2	64	256	256	**50%**	100%	**30%**	6bit	197,184	77,581
conv10	13x13x1000	1x1/1 (x1000)	1					**20%** (3x3)		6bit	513,000	103,400
avgpool10	1x1x1000	13x13/1	0									
	activations			parameters			compression info				1,248,424 (total)	**421,098** (total)

Nota. Dimensiones arquitectónicas de *SqueezeNet*. Tomado de *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size* (p. 6), por Iandola et al. (2016), arXiv.

d. *DenseNet*

DenseNet tiene tres bloques densos que tienen cada uno la misma cantidad de capas. Antes de ingresar al primer bloque denso, se realiza una convolución con 16 canales de salida en las imágenes de entrada. Para las capas convolucionales con un tamaño del núcleo de 3 × 3, cada lado de las entradas se rellena con ceros en un píxel para mantener fijo el tamaño del mapa de características. Utiliza una convolución de 1 × 1 seguida de una agrupación promedio de 2 × 2 como capas de transición entre dos bloques densos contiguos. Al final del último bloque denso, se lleva a cabo una agrupación promedio global y, a continuación, se añade un clasificador *softmax*. Los tamaños de los mapas de características en los tres bloques densos son 32 × 32, 16 × 16 y 8 × 8, respectivamente. Experimentan con la estructura básica de *DenseNet* con configuraciones {L = 40, k = 12}, {L = 100, k = 12} y {L = 100, k = 24}. Para *DenseNetBC* se evalúan las redes con configuraciones {L= 100, k= 12}, {L= 250, k= 24} y {L= 190, k= 40} (Huang et al., 2017). En experimentos con *ImageNet*, utiliza una estructura *DenseNet-BC* con 4 bloques densos en imágenes de entrada de 224×224. La capa de convolución inicial comprende 2k convoluciones de tamaño

7×7 con paso 2; el número de mapas de características en todas las demás capas también se deriva de la configuración de k. Las configuraciones de red exactas que utilizan en *ImageNet* se muestran en la Figura 9.

Figura 9

Arquitecturas DenseNet para ImageNet

Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6	[1 × 1 conv; 3 × 3 conv] × 6
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12	[1 × 1 conv; 3 × 3 conv] × 12
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	[1 × 1 conv; 3 × 3 conv] × 24	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 48	[1 × 1 conv; 3 × 3 conv] × 36
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	[1 × 1 conv; 3 × 3 conv] × 16	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 32	[1 × 1 conv; 3 × 3 conv] × 24
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

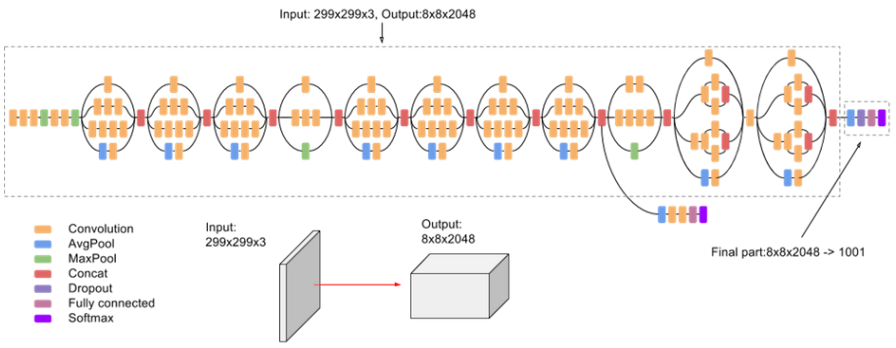
Nota. La tasa de crecimiento para todas las redes es k = 32, cada capa "conv" corresponde a la secuencia *BN-ReLU-Conv*. Tomado de *Densely connected convolutional networks* (p. 4), por Huang et al. (2017), arXiv.

e. *Inceptionv3*

Inceptionv3 tienen un costo de computación relativamente modesto en comparación con arquitecturas más simples y monolíticas. Nuestra versión de más alta calidad de *Inceptionv3* alcanza un 21.2 %, top-1 y un 5.6 % top-5 de error para la evaluación de un solo corte en la clasificación *ILSVR* 2012, estableciendo un nuevo estado del arte (Szegedy et al., 2016).

Figura 10

Arquitectura Inceptionv3



Nota. Arquitectura Inceptionv3. Tomado de Comparative analysis of deep

learning convolutional neural networks based on transfer learning for pneumonia detection (p. 1167), Chiwariro y Wosowei (2023), IJRASET.

B.1.2 Métricas de evaluación en la clasificación

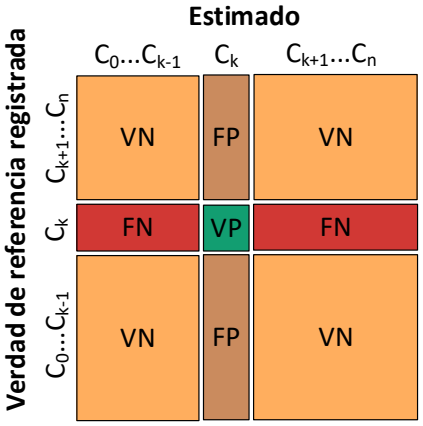
Cuando se trata de problemas de aprendizaje automático, encontrará muchos tipos diferentes de métricas en el mundo real (Thakur, 2020). A la hora de trabajar con los clasificadores, un aspecto fundamental es la definición de un criterio que permita comparar el desempeño de cada uno para establecer cuál de ellos es mejor que otro al enfrentarse a un determinado problema (Arostegui, 2022).

a. Matriz de confusión

Una matriz de confusión no es más que una tabla de VP, FP, VN y FN. Con la matriz de confusión, puede ver rápidamente cuántas muestras se clasificaron incorrectamente y cuántas se clasificaron correctamente (Thakur, 2020). P (condición positiva) indica el número de casos positivos reales en el conjunto de datos. N (condición negativa) representa el número de casos negativos reales en el conjunto de datos. A continuación, observamos cómo se presenta una matriz de confusión de una clasificación con n clases.

Figura 11

Matriz de confusión para clasificación multiclase



Nota. Adaptado de *Activity, Context, and Plan Recognition with Computational Causal Behaviour Models* (p. 71), por Krüger (2016), *University of Rostock*.

En la Figura 11 al considerar la clase k $(0 \leq k \leq n)$, se pueden obtener cuatro resultados de clasificación diferentes:

- VP Verdadero Positivo (verde): Es un resultado de la prueba que confirma correctamente la presencia de una condición o característica.
- VN (Verdadero Negativo (anaranjado): Es un resultado de la prueba que confirma correctamente la ausencia de una condición o característica.
- FP Falso Positivo, Error Tipo I (marrón): Es un resultado de prueba que indica erróneamente que una condición o atributo en particular está presente.
- FN Falso Negativo, Error Tipo II (rojo): Es un resultado de la prueba que incorrectamente señala la ausencia de una condición o atributo específico.

A partir de estos conceptos se muestra la métrica de evaluación exactitud basada en la matriz de confusión.

b. Exactitud

Relación entre las predicciones correctas sobre el total de instancias.

$$ACC = \frac{VP + VN}{P + N} = \frac{VP + VN}{VP + VN + FP + FN}$$

B.1.3 Descenso de gradiente estocástico (*SGD*)

El descenso de gradiente estocástico y sus variantes son los algoritmos más utilizados para el entrenamiento de modelos de aprendizaje profundo. Las iteraciones matemáticas del *SGD* se muestran a continuación:

$$W_k = W_{k-1} - \alpha_{k-1} \nabla f(W_k - 1)$$

dónde, $W_k = K^{th}$ iteración y α_k representa la tasa de aprendizaje (Habib y Qureshi, 2022).

1.2 Antecedentes

1.2.1 Internacionales

Shams et al. (2020) presentan un algoritmo para reconocer letras y caracteres árabes basado en el uso de *DCNN* y *SVM*, abordan el problema de reconocer los caracteres escritos a mano en árabe determinando la similitud entre las plantillas de entrada y las plantillas prealmacenadas, alcanza una exactitud de 95.07% de *CRR* y un 4.93% de *ECR*.

Gurav et al. (2020) trabajan con un conjunto de datos de 34604 imágenes escritas a mano; incorporan una *DCNN* para extraer características y clasificar las imágenes de entrada, en este proceso utilizan capas convolucionales consecutivas, lo que aporta una ventaja adicional en el proceso de extracción de características de nivel superior; su modelo entrenado demuestra una exactitud del 99.65%.

Agrawal y Awasthi (2021) mencionan que el rendimiento de las redes neuronales profundas son sobresalientes en comparación a otras técnicas, y se utilizan principalmente el campo del reconocimiento de patrones como: dígitos escritos a mano, caracteres, imágenes, rostros, sonido, habla, entre otros; logran una exactitud del 98.70% en las pruebas y del 99.76% en el entrenamiento para las muestras de *ARDIS*, también logran una exactitud del 98.22% en el entrenamiento y del 93.01% en las pruebas con el conjunto de datos de muestras de *USPS*.

Aneja y Aneja (2019) presentan un análisis de modelos previamente entrenados para reconocer alfabetos devanagari escritos a mano mediante el aprendizaje por transferencia para *DCNN*; implementan *AlexNet*, *DenseNet*, *VGG* e *Inceptionv3*, siendo este último el que funciona mejor en términos de exactitud, logrando una exactitud del 99% con un tiempo de época promedio de 16.3 minutos, mientras que *AlexNet* funciona más rápido con 2.2 minutos por época en promedio y logra una exactitud del 98%.

Rajpal y Garg (2023) indican que los modelos de *DCNN* han tenido éxito en la resolución de problemas de reconocimiento de patrones, pero a expensas de una cantidad considerable de parámetros entrenables y grandes cargas

computacionales; su modelo alcanzó con éxito una exactitud de reconocimiento del 99.20%.

Pearlsy y Sankar (2023) mencionan que la técnica de última generación que utiliza *DCNN* exige una gran cantidad de conjuntos de datos etiquetados; tienen como objetivo desarrollar un modelo de red *CNN* previamente entrenado para reconocer caracteres escritos a mano en malayalam, utilizando un conjunto de datos de tamaño pequeño; obtienen una precisión de prueba del 78.05% con *ResNet50* para imágenes binarias; aunque *ResNet50* está previamente entrenado utilizando imágenes en color.

Jose y Pushpalatha (2023) utilizan *DCNN* para el reconocimiento de caracteres escritos a mano en malayalam mediante el aprendizaje por transferencia; entrenándolos en 15 épocas, *Inception-V4* logró una exactitud del 99% con un tiempo de época promedio de 16.3 minutos, al mismo tiempo, *AlexNet* logró una exactitud del 98% con un tiempo de entrenamiento promedio de 2.2 minutos por época; utilizan conexiones residuales dentro de la arquitectura *Inception* tradicional, lo que resultó en un rendimiento de aprendizaje de última generación con la exactitud más alta del 99.69% y un tiempo de época promedio de 15.1 minutos.

Masruroh et al. (2023) precisan que el aprendizaje por transferencia es un modelo que ha sido entrenado por conjuntos de datos anteriores a otros conjuntos de datos y es adecuado para su uso en modelos con conjuntos de datos pequeños porque puede mejorar la exactitud del modelo; algunas de las arquitecturas *CNN* utilizadas en este estudio para crear modelos de reconocimiento de escritura árabe son *ResNet*, *DenseNet*, *VGG16*, *VGG19*, *Inceptionv3* y *MobileNet*. El mejor modelo para clasificar los conjuntos de datos *AHCD* e Hijja es *VGG16* con optimizador Adam y tasa de aprendizaje de 0.0001 con una exactitud 95.67% y 93.59% respectivamente.

Ansari et al. (2022) proponen cómo se puede utilizar el aprendizaje por transferencia en el reconocimiento de caracteres escritos a mano en Devanagari; utilizan una *DCNN* para la extracción y clasificación de características, donde las características extraídas por la *DCNN* se envían a *SVM*; logran una precisión del

99.41% en el conjunto de prueba, también reducen la cantidad de parámetros utilizados para la clasificación.

Riaz et al. (2022) se centran en la evaluación de modelos de aprendizaje profundo en un conjunto de datos compilado de manuscritos de dígitos en urdu; la evaluación lo realizan para algoritmos *DCNN* como *VGGNet16*, *Inceptionv3*, *ResNet50* y *DenseNet121*; sus modelos convolucionales están previamente entrenados en *ImageNet*; la exactitud de las pruebas se observan como *ResNet50* con 96%, *Inceptionv3* con 95%, *VGGNet16* con 95% y *DenseNet121* con 94%.

Awni et al. (2022) examinan el rendimiento de tres *DCNN* que se han inicializado aleatoriamente para reconocer palabras manuscritas en árabe; evalúan el rendimiento del modelo *ResNet18* que ha sido previamente entrenado en el conjunto de datos *ImageNet* para la misma tarea, y logran una exactitud de reconocimiento hasta del 96.11%, lo que representa una mejora de casi el 2.5% en comparación con otros métodos de última generación.

Sikder et al. (2022) entrenan el conjunto de caracteres bengalí utilizando *DCNN* con más de 360 caracteres distintos entre los cuales hay muchas similitudes entre los diferentes caracteres, usan un conjunto de datos de imágenes a la que pertenecen 84 clases distintas con la capacidad de detectar letras bengalíes escritas a mano individuales, incluidos dígitos y vocales, consonantes y caracteres compuestos; su modelo logra una exactitud del 94.73%.

Deore y Pravin (2020) proponen el enfoque de ajuste y el análisis de la *DCNN* para la clasificación de caracteres escritos a mano en Devanagari; el conjunto de datos con el que investigan consta de un total de 5800 imágenes aisladas de 58 clases de caracteres únicos: 12 vocales, 36 consonantes y 10 números; su primer modelo *VGG16* logra una exactitud de prueba del 94.84% con una pérdida de entrenamiento del 0.18; el segundo modelo *DHCRS* logra una exactitud de prueba del 96.55% con una pérdida de entrenamiento del 0.12.

T. Ghosh et al. (2021) utilizan arquitecturas *DCNN* previamente entrenadas de última generación para clasificar 231 caracteres escritos a mano en bengalí diferentes utilizando el conjunto de datos *CMATERdb*; después de 50 épocas, *InceptionResNetV2* logra la mejor exactitud de 96.99%. *DenseNet121* e

InceptionNetV3 también proporcionaron una exactitud de reconocimiento notable de 96.55% y 96.20%, respectivamente; también consideran una combinación de arquitecturas entrenadas *InceptionResNetV2*, *InceptionNetV3* y *DenseNet121* que proporcionaron una mejor exactitud de reconocimiento del 97.69%.

Elleuch et al. (2021) comparan los modelos basados en *DCNN* con el aprendizaje desde cero versus la estrategia de aprendizaje por transferencia; así, los modelos *ResNet* y *VGG* con *transfer learning* logran resultados con una precisión del 98.99% y 98.10% respectivamente, aplicados a un conjunto de datos de imágenes de la palabra manuscrita árabe *IFN/ENIT*.

Rajpal et al. (2021) en su modelo propuesto abordan el problema del reconocimiento de caracteres del idioma oficial de la India el hindi escrito a mano, utilizan modelos *DCNN* previamente entrenados, a saber *Inceptionv3-Net*, *VGG19-Net* y *ResNet50* para extraer características destacadas de las imágenes de los caracteres; su modelo logró puntuaciones significativas de *precision*, *recall* y *F1-score*, con tasas de 98.78%, 98.67% y 98.69% respectivamente, y una exactitud de reconocimiento general del 98.73%.

Tuncer et al. (2022) hablan de la fase de extracción profunda de características, en el cual seleccionan 13 *CNN*s preentrenados, su método de clasificación alcanzó una exactitud de 97.16%. en los datos recopilados de firmas manuscritas y del 100% en el conjunto de datos *CEDaR*.

Rajpal y Garg (2023b) precisan que debido a la abundancia de curvas y formas parecidas de los símbolos, existe un nivel de dificultad en el reconocimiento de los números Devanagari; y que el método de bajo costo de clasificación sugerido para obtener características precisas a partir de imágenes numéricas son los modelos de aprendizaje profundo *VGG-16Net*, *VGG-19Net*, *ResNet-50* e *Inception-v3* para abordar estos problemas; utilizan el método de reducción de dimensionalidad; y logran una exactitud de reconocimiento del 99.72%.

Rani et al. (2020) utilizan el conocimiento de un gran corpus de datos del sistema de reconocimiento Devanagari como datos de entrenamiento para realizar el reconocimiento de caracteres kannada escritos a mano que tienen un corpus de

datos más pequeño; la transferencia de conocimiento para el reconocimiento lo realizan mediante arquitectura de red de aprendizaje profundo *VGG19*, después de la evaluación en 10 épocas registran una exactitud del 73.51% con una función de pérdida del 16.18%.

Sutramiani et al. (2021) realizan el reconocimiento de caracteres balineses en hojas de palma o manuscrito *lontar*, combinan el umbral gaussiano adaptativo y el codificador automático convolucional para el aumento de datos; según experimentos con *InceptionResnetV2*, *DenseNet169*, *ResNet152V2*, *VGG19* y *MobileNetV2*, su método propuesto logra una mejor exactitud del 96.29%.

Purnamawati et al. (2018) usan las *DCNN* en el proceso de reconocimiento de imágenes de caracteres coreanos en función del modelo que ha sido entrenado, como el modelo Inception-v3, posteriormente, llevan a cabo un proceso de reentrenamiento utilizando la técnica de *transfer learning* con el valor del modelo entrenado y reentrenado para desarrollar un nuevo modelo con un mejor rendimiento sin errores sistémicos específicos, obtienen una exactitud de las pruebas del 86.9%.

Alsaeedi et al. (2018) utilizan una *CNN* para el reconocimiento de caracteres y una *TNN* para la lectura de palabras, debido a que la segmentación de caracteres árabes mencionan que es un paso muy complicado, reconocen solo el primero, el último carácter de todos los componentes conectados de la palabra reconocida; la evaluación lo realizan en una base de datos de imágenes de letreros de nombres de ciudades impresos y la tasa de reconocimiento es del 98%.

Garg et al. (2022) describen los manuscritos indios a partir de sus imágenes escaneadas utilizando *CNN*; su método utiliza principalmente procesamiento de imágenes para aprovechar las características visuales imágenes y las clasifica según la diferencia en los estilos de escritura en términos de trazos y formación de letras, restringimos la consideración a seis manuscritos en lengua india escritos entre los siglos XVI y XX. Su modelo supera a otras arquitecturas conocidas y proporciona más del 90% y 80% de exactitud en los datos de entrenamiento y validación, respectivamente.

Sandhya y Geetha (2022) proponen un modelo de reconocimiento de caracteres basado en *DenseNet121* que reconoce eficazmente los caracteres kannada escritos a mano; utilizan *transfer learning* para mejorar el rendimiento general del modelo. Su modelo propuesto logró una precisión de entrenamiento del 96.7% y una precisión de prueba del 96.28%, lo que demuestra su eficacia.

Chandrakala y Thippeswamy (2020) presentan un enfoque para el reconocimiento de caracteres de manuscritos históricos escritos a mano en Kannada; mencionan que una *DCNN* es un modelo que unifica la extracción y clasificación de características; experimentan en la digitalización de inscripciones históricas en piedra en Kannada que pertenecen al siglo XI, llegando a una exactitud del 70% con el método de clasificación *AlexNet*.

Sridevi y Rangarajan (2021) proponen una *DCNN* para el reconocimiento de caracteres de algunos textos antiguos en kannada afectados por diversas degradaciones que resultan en roturas y dilataciones de los caracteres que presentan desafíos en el proceso de reconocimiento; utilizan 156 clases de caracteres, cada clase con 100 instancias. El rendimiento lo evalúan durante 4 épocas con 60 iteraciones por época, e informan de una exactitud de clasificación más alta del 99.51% para el entrenamiento.

Agrawal et al. (2021) se centran en el campo del reconocimiento de dígitos escritos a mano para la clasificación de patrones utilizando el conjunto de datos *MNIST*, que es una colección de 70000 imágenes; el modelo *CNN* que proponen logra el 99.06% de exactitud de entrenamiento y el 98.80% de exactitud de prueba en la época 10.

Nanehkaran et al. (2021b) se centran en la detección de dígitos escritos a mano en farsi el cual se utiliza ampliamente en la mayoría de los contextos que implican la recopilación de información numérica digital genérica, como la lectura de cheques o dígitos de códigos postales; su mejor exactitud basados en todas las características que investigan es 99.45%.

Zhong et al. (2015) aplican la agrupación múltiple y el aumento de datos con transformación no lineal a una *CNN* para caracteres chinos impresos de múltiples fuentes; para clasificar 3755 clases de caracteres chinos impresos en 280

fuentes muy diversas y 120 fuentes seleccionadas manualmente, logran tasas de reconocimiento del 94.38% y 99.74% en el primer y último caso, respectivamente, lo que indica la efectividad de los métodos propuestos.

Wu et al. (2017a) evalúan la clasificación y sobre segmentación de caracteres en el reconocimiento de textos escritos a mano en chino; precisan que las *CNN* mejoran significativamente el rendimiento tanto en el conjunto de datos de escritura china *CASIA-HWDB* como en *ICDAR-2013*; su tasa de exactitud y tasa de corrección a nivel de carácter es del 95.88% y el 95.95%, respectivamente.

Zhuang et al. (2021) indican que en las investigaciones hacia la visión por computadora y el reconocimiento de patrones, los métodos que se basan en *CNN* han mostrado ventajas en el reconocimiento de caracteres chinos escritos a mano, en su modelo las imágenes de entrada se preprocesan mediante filtrado mediano para suavizar y reducir el ruido, sus resultados experimentales muestran que la tasa de exactitud del reconocimiento se acerca al 90.91% después de 5000 entrenamientos, el error cuadrático medio se reduce a 0.0079; su sistema tiene un buen rendimiento en pruebas en tiempo real.

Altwaijry y Al-Turaiki (2021) presentan un nuevo conjunto de datos de letras árabes escritas exclusivamente por niños de entre 7 y 12 años denominado Hijja, que contiene 47434 caracteres, escritos por 591 participantes, además, proponen un modelo de reconocimiento automático de escritura basado en *CNN*; sus resultados muestran que el rendimiento de su modelo es prometedor, logrando la exactitud de 97% y 88% en el conjunto de datos *AHCD* y el conjunto de datos Hijja, respectivamente.

Inunganbi et al. (2021) presentan un sistema de reconocimiento de la escritura a mano Meitei Mayek o escritura Manipuri; utilizando una *CNN*, el reconocimiento de caracteres lo realizan utilizando imágenes de los caracteres a escala de grises, sus experimentos lo llevan a cabo con 14700 imágenes de muestra y obtienen una tasa de reconocimiento del 98.70%.

Pratama (2021) menciona que las *CNN* ha demostrado ser poderoso para la clasificación de imágenes, y que la arquitectura *CNN* que tiene alta precisión para ese problema es *LeNet-5*; el autor trabaja con el conjunto de datos de escritura

a mano *MNIST* e implementa la *SVD* para la extracción de características como método de preprocesamiento; precisa que el modelo *LeNet-5* proporciona una exactitud del 99.03%, además, indica que la relación de compresión con 2 componentes puede reducir aproximadamente 7.87 veces el tamaño del archivo.

Laurencich-Minelli (2016b) indica que existen dos documentos jesuíticos hallados en un archivo italiano, donde narran sobre los quipu de los Incas como medio de comunicación. Se vislumbra un sofisticado sistema de escritura fundamentado en principios numéricos, que un grupo de jesuitas intentó reintroducir durante los primeros años de la colonia española. Este sistema incluye el quipu, destacándose por sus diversas formas y funciones: desde los quipus numéricos hasta los literarios, utilizados en una compleja contabilidad.

Zavala et al. (2021) señalan que, debido a su función mnemotécnica, los quipus son conocidos como "nudos que hablan". Estos sistemas de registro visual-táctil consistían en hilos de fibra de llama, alpaca o algodón, que eran teñidos y entrelazados mediante una compleja secuencia de nudos. Según crónicas coloniales tempranas, se usaban para recopilar datos y mantener registros sobre obligaciones tributarias, censos de población, información calendárica, producción y organización militar, así como para narrar hazañas épicas. Aunque se ha acumulado conocimiento sobre ellos, el quipu como sistema de comunicación mnemotécnica incaica sigue planteando más preguntas que respuestas en cuanto a sus posibles lecturas e interpretaciones.

Animato et al. (1989) en su manuscrito describen 17 símbolos del manuscrito de Oliva con su significado que en cierta medida recuerdan a los *tocapu* de las tablas de Guamán Poma de Ayala y que encabezarían los hilos que salen de la cuerda principal de aquellos quipus particulares que expresan conceptos en lugar que números.

Cabral (2019) menciona que la lengua representa de forma cristalizada la cultura de una determinada sociedad. E indica que la lengua quechua tiene alrededor de cinco mil años y se originó en Perú; durante el imperio Inca, el quechua se estableció como la lengua principal de comunicación entre personas que hablaban diferentes lenguas y se consolidó como lengua de administración y control; está relacionada con la historia y la memoria de un pueblo y está

intrínsecamente relacionada con la transmisión de conocimiento, la cosmovisión y su transmisión a las generaciones futuras, por lo que es importante preservar el quechua, con el fin de preservar y transmitir los conocimientos ancestrales de los pueblos andinos.

Silverman (2011) basándose en datos etnográficos proporcionados por los cronistas, decodificó varios motivos Incas conocidos como *tocapu* que se refieren a la tecnología agrícola, identificó nombres singulares y plurales, adjetivos, y dos sufijos para mostrar que los Incas usaban la escritura pictográfica, que proviene de formas de la naturaleza u objetos concretos hechos por el hombre, para fijar su sabiduría antes de la llegada de los españoles.

Bouysse-Cassagne y Harris (1988) mencionan que los Incas poseían genuinas formas de memorización, aunque de concepción distinta a las formas occidentales de escritura: como quipu, *qillqa* (cántaros dibujados) y tejidos.

Sandron (1999) presenta la lista de los ideogramas proporcionados por la *Nueva Crónica y Buen Gobierno* e *Historia et Rudimenta Linguae Piruanorum*; en su estudio precisa “la existencia de una escritura pictográfica, ideográfica e ideográfica-silábica quechua-inca” que tendría “varios niveles de lectura”.

Ziótkowski y Siemianowska (2021) Destacan la existencia de una tradición Inca de representaciones pictóricas narrativas para registros históricos, respaldada por arte prehispánico y fuentes escritas. Las “tablas de *Poquen Cancha*” son la referencia más conocida en relación con las “pinturas históricas” de los Incas. Aunque hoy están desaparecidas, estas tablas se usaron y consultaron hasta al menos mediados de 1560. Se sugiere que, durante el periodo colonial, estas “pinturas históricas” podrían haber sido parcialmente reproducidas en diversos soportes, como lienzos y keros.

Frame (2010) examina las expresiones anidadas de la organización social y política en las túnica de *tocapu* que lleva el Inca en los dibujos de Guamán Poma, presenta datos etnohistóricos, etnográficos y arqueológicos; de codificación genuina, contextualizada en una escena compleja.

Zuidema (2014) menciona que si bien los quipus fueron utilizados en todas partes del imperio, su difusión responde probablemente a intereses de la

administración política y ritual Inca, aunque hemos avanzado en el conocimiento técnico y teórico sobre su uso, sobre las cosas que, en general, registraron y sobre algunos temas específicos que se documentaron, todavía se puede encontrar la opinión, en la literatura reciente, que no se ha "descifrado" el contenido de ningún quipu y menos el de uno narrativo.

Eeckhout y Danis (2004) analizaron detalladamente 116 de las 130 tablas y tomaron en cuenta 36 *tocapu*, de los cuales 34 diseñados en las túnicas de los Sapa Incas, muchos de los cuales, identificados con sus nombres, y de sus Coya. Enumeraron 230 *tocapu*. De hecho, definir su número es subjetivo, pues depende de lo que consideramos variaciones significantes e insignificantes.

Gentile (2008) indica que el registro andino de datos y su comunicación están asociados a un objeto, el quipu, que los Incas perfeccionaron y potenciaron. Sin embargo, existen dibujos que persisten conservando su forma a través de diferentes soportes y estilos, lo que sugiere la comunicación de una idea con su manifestación material. La autora analiza y contextualiza uno de estos dibujos en términos de espacio y tiempo, y ofrece una interpretación de su significado.

Cummins (2011) menciona que para el quipu, la forma y objeto Inca más intensamente estudiado, sólo podemos describir, cómo la información podría haber sido codificada a través de un conjunto recursivo de notaciones posicionales decimales y relaciones binarias. No podemos decodificar ese sistema. para la información que se supone que contiene cualquier quipu existente. Es decir, se supone que los ejemplos discretos del quipu contienen información específica, ya sea estadística, histórica o religiosa, pero no se pueden obtener detalles específicos.

Hyland y Hyland (2020) en su artículo analizan dos epístolas de quipu de los Andes centrales, y describen el sistema de comunicación logosilábico que representan un sistema fonético.

1.2.2 Nacionales

Roncoroni (2017) precisa que los quipus son sistemas de cuerdas y nudos de diferentes formas y colores utilizados hace cientos de años por las culturas peruanas prehispánicas como sistemas de información, almacenamiento de datos

y para realizar algún tipo de operaciones matemáticas; algunos autores también consideran al quipu como un sistema de escritura, utilizado para registrar poemas y acontecimientos sociales y militares importantes; muestra 40 palabras primarias y su significado en inglés, 76 sílabas, y parte del poema estudiado por las investigaciones de *di Sangro*; ofrecen conocimientos históricos interesantes sobre procesos computacionales, como las estructuras de datos, y conocimientos originales sobre escritura y gramáticas generativas.

De la Jara (2017) estudiando las palabras de los viejos vocabularios de la lengua de los Incas, y comparándolas con las descripciones de los antiguos relatos, descubrí que coincidían al referirse a una escritura multicolor, con signos en forma cuadrada y a textos sobre tejidos, madera y otros materiales. Identificó los signos cuadrados en el material arqueológico, y sus catálogos y estudios estructurales confirmaron la existencia de escritura incluso durante la dominación española.

Huillca-Huallpa (2016) menciona que los Incas tuvieron una cultura muy desarrollada, porque tuvieron una gran capacidad creadora, fina inteligencia para analizar y simplificar el mensaje universal en su escritura que está basada en diversas formas, trazos, códigos, figuras y colores estéticos; la escritura simbólica tiene representación pictográfica, ideográfica, geométrica, artística y abstracta. En su estudio muestra 134 símbolos base de los quipus de escritura.

CAPÍTULO II

PLANTEAMIENTO DEL PROBLEMA

2.1 Identificación del problema

El estudio de los quipus de escritura se enfrenta a una notable carencia de comprensión integral, a pesar de los avances en la interpretación de los quipus como sistema de registro numérico, se carece de un entendimiento definitivo de su función más allá de la representación de números, además, incluyendo el hipotético quipu de palabras, sigue siendo en gran medida enigmático y su naturaleza y alcance son poco claros.

Con el estudio se entiende, fundamenta y evidencia los quipus de escritura, a fin de llenar un vacío de conocimiento y también se describe los posibles patrones significativos y recurrentes en su sistema de escritura, que están presentes en los documentos escritos como quipus de escritura, crónicas, tablillas de madera, tablones, tejidos, entre otros objetos; para lo cual se utilizan algoritmos de procesamiento de imágenes y la visión computacional con la finalidad de identificar el significado de cada una de las imágenes que representan palabras u oraciones base.

El quechua es una familia lingüística hablada a lo largo de la cordillera de los Andes de Sudamérica, el número de quechua hablantes se estima entre ocho y nueve millones desde el sur de Colombia hasta el norte de Argentina, así como en varios puntos de la selva de Colombia, Ecuador y Perú (Steckbauer, 2000); los resultados que se busca obtener en este estudio permiten comprender los quipus de escritura para la población quechua hablante.

En lo académico se demuestra que la cultura precolombina Inca si tuvo escritura propia, y esta tesis sirve como base para investigadores científicos en los temas como los quipus de escritura y las escrituras precolombinas.

La importancia se basa en descifrar la escritura, no sólo la Inca o precolombina, sino también en otros idiomas y dialectos a través de la visión computacional, además, en proponer su uso entre la población quechua hablante; para lo cual se utiliza el procesamiento de las imágenes como la segmentación, adquisición de la imagen, preprocesamiento, extracción de rasgos, y clasificación.

Para obtener el mejor resultado se ha probado los modelos *AlexNet*, *VGG11-BN*, *SqueezeNet*, *DenseNet121, e Inceptionv3*, basado en la métrica exactitud, el que tiene mejor resultado es el indicado para el reconocimiento de la simbología base de los quipus de escritura.

2.2 Enunciados del problema

2.2.1 Problema general

- ¿Cómo desarrollar un sistema de red neuronal convolucional profunda que obtenga mejores resultados en el reconocimiento automático de la simbología base de los quipus de escritura?

2.2.2 Problemas específicos

- ¿Cómo es el sistema de escritura Inca, particularmente los quipus de escritura y determinar sus posibles patrones?
- ¿Cómo construir un conjunto de datos significativo de imágenes de los quipus de escritura que permita entrenar modelos mediante aprendizaje por transferencia?
- ¿Cuál es el mejor modelo de redes neuronales convolucionales entrenado evaluado bajo las mismas condiciones y métricas en la tarea del reconocimiento de la simbología base de los quipus de escritura?

2.3 Justificación

En base a los resultados de este estudio nos permite entender y descifrar el sistema de escritura en las culturas incaicas siendo este una brecha en nuestro entendimiento de cómo se transmitía y registraba la información en esta sociedad, además, busca llenar este vacío, proporcionando una visión más completa de las prácticas comunicativas y de registro de los quechua hablantes.

Con este estudio se entiende, fundamenta y evidencia la simbología base utilizada en el quipu de escritura y que no solo tiene implicaciones académicas, sino también culturales e históricas significativas. Proporciona una mayor apreciación de la riqueza cultural y la complejidad social de la civilización incaica, así como de su sistema de transmisión de conocimiento y su organización sociopolítica.

Este estudio contribuye a validar o refutar hipótesis existentes y proporcionar una base más sólida para futuras investigaciones. Una comprensión más profunda de los quipus de escritura tiene implicaciones educativas y contribuye a la difusión de la cultura incaica. Nos proporciona materiales valiosos para la enseñanza y la divulgación cultural, tanto en entornos académicos como en museos y otros espacios educativos.

La propuesta de esta investigación se centra en descifrar los quipus de escritura, identificando sus características y patrones distintivos. Se construye un conjunto de datos significativo de imágenes que representan la simbología base, seguido por el entrenamiento de cinco modelos de redes neuronales convolucionales preentrenados. Este enfoque no solo permite identificar las formas de palabras o frases, sino también avanzar en el desarrollo de un algoritmo eficaz para el reconocimiento automático de los quipus de escritura.

2.4 Objetivos

2.4.1 Objetivo general

- Desarrollar un sistema de red neuronal convolucional profunda que obtenga mejores resultados en el reconocimiento automático de la simbología base de los quipus de escritura.

2.4.2 Objetivos específicos

- Descifrar el sistema de escritura Inca, particularmente los quipus de escritura y sus posibles patrones.
- Construir un conjunto de datos significativo de imágenes de los quipus de escritura que permita entrenar modelos mediante aprendizaje por transferencia.
- Evaluar los modelos de redes neuronales convolucionales entrenados bajos las mismas condiciones y métricas; y determinar el mejor modelo en la tarea de reconocimiento de la simbología base de los quipus de escritura.

2.5 Hipótesis

2.5.1 Hipótesis general

- El desarrollo de un sistema de red neuronal convolucional profunda permite obtener mejores resultados en el reconocimiento automático de la simbología base de los quipus de escritura, superando la exactitud de otros métodos.

2.5.2 Hipótesis específicas

- El sistema de escritura Inca se basa en los quipus de escritura que es simbólica, pictográfica, ideográfica, geométrica, abstracta, logosilábica y cifrada; tiene patrones muy definidos.
- La construcción del conjunto de datos significativo de imágenes de los quipus de escritura permite entrenar modelos mediante aprendizaje por transferencia de manera efectiva.
- El modelo de red neuronal convolucional profunda *DenseNet121* mejora significativamente la exactitud en el reconocimiento automático de la simbología base de los quipus de escritura en comparación con otros métodos.

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1 Lugar de estudio

Este estudio se realizó en la ciudad de Puno – Perú, ubicado a 3820 m.s.n.m. en el cual se creó el conjunto de datos de imágenes obtenidos específicamente de la región quechua hablante de los andes de Sudamérica, el desarrollo y entrenamiento del modelo se realizó en un ambiente virtual, específicamente en la plataforma en la nube *Google Colaboratory*, con el uso del entorno de ejecución *GPU*.

3.2 Población

El presente estudio de investigación se enfocó en las imágenes que tienen relación con la simbología base de la escritura Inca, específicamente con los quipus de escritura. Por lo que la población estuvo conformada por 167 clases o palabras/frases (símbolos base).

3.3 Muestra

Se consideró para la muestra de la investigación el muestreo por conveniencia, donde se seleccionó 40 clases o palabras/frase (símbolos base), debido a que estas clases fueron accesibles y representativas para su recopilación y estudio; que corresponden al 23.95% del total de símbolos identificados hasta el momento de esta investigación.

3.4 Método de investigación

La investigación fue aplicada de enfoque, con un nivel explicativo, utilizando datos cuantitativos y centrada en aplicaciones prácticas, con un diseño no experimental, de corte longitudinal y en base a los resultados se determinó la relación con reconocimiento automático de la simbología base, y se enmarcó dentro de la corriente epistemológica positivista debido a que se analiza a los quipus de escritura desde una perspectiva matemática y cuantitativa.

3.5 Descripción detallada de métodos por objetivos específicos

3.5.1 Método del objetivo específico 1

En este objetivo específico de estudio, se consideró la estructura del quipu de escritura como la variable independiente y los patrones identificados como la variable dependiente. Se utilizaron diversos materiales y recursos, incluyendo quipus, *qhapaq khipus*, tablillas, formas trabajadas en metales como oro, plata y cobre; en madera como tabillas y tablones; en tejidos como algodón o fibra de camélidos, trenzados con entrelazado de hilos; en *tocapu* en uncus, en *urpu*, en keros en tapicería de cumbre y otros objetos representativos de la escritura precolombina. También se consultaron fuentes históricas como la *Lettera apologetica dell'esercitato Accademico della Crusca* de Raimondo di Sangro , el *Qhapaq Khipu Teqsi Simi* de Huillca-Huallpa (2016), el *Vocabulario en la Lengua General del Perú llamada Quichua* de Ricardo (1586) y el *Vocabulario de la lengua general de todo el Perú llamada lengua Qquichua o del Inca* de Gonzalez-Holguin (1608). Para la recopilación de imágenes se utilizaron dispositivos como un *smartphone Xiaomi Mi A3* con una cámara de 48 megapíxeles, además de las redes sociales y buscadores en internet. Para el análisis de datos se aplicó la prueba estadística análisis descriptivo, en *Python* 3.6.0 utilizando una portátil con procesador *Intel® Core™ i7-8550U* y tarjeta gráfica *Intel® UHD Graphics 620*.

3.5.2 Método del objetivo específico 2

En este objetivo específico de estudio, la variable independiente fue un conjunto de datos significativo de imágenes de quipus de escritura, mientras que la variable dependiente fue la eficacia del entrenamiento de los modelos mediante aprendizaje por transferencia. Se utilizaron diversos materiales, incluidos un conjunto de datos de imágenes de 40 clases con 4000 imágenes de las formas de los quipus de escritura, *Python* en la versión 3.8.10, y *Google Colaboratory*; y se propuso un método para el proceso de desarrollo de un sistema de reconocimiento de objetos o imágenes mediante aprendizaje por transferencia para tareas de visión por computadora, como se ilustra en la Figura 12. Respecto al análisis de datos, se aplicaron métodos de aprendizaje automático para identificar patrones y hacer predicciones.

3.5.3 Método del objetivo específico 3

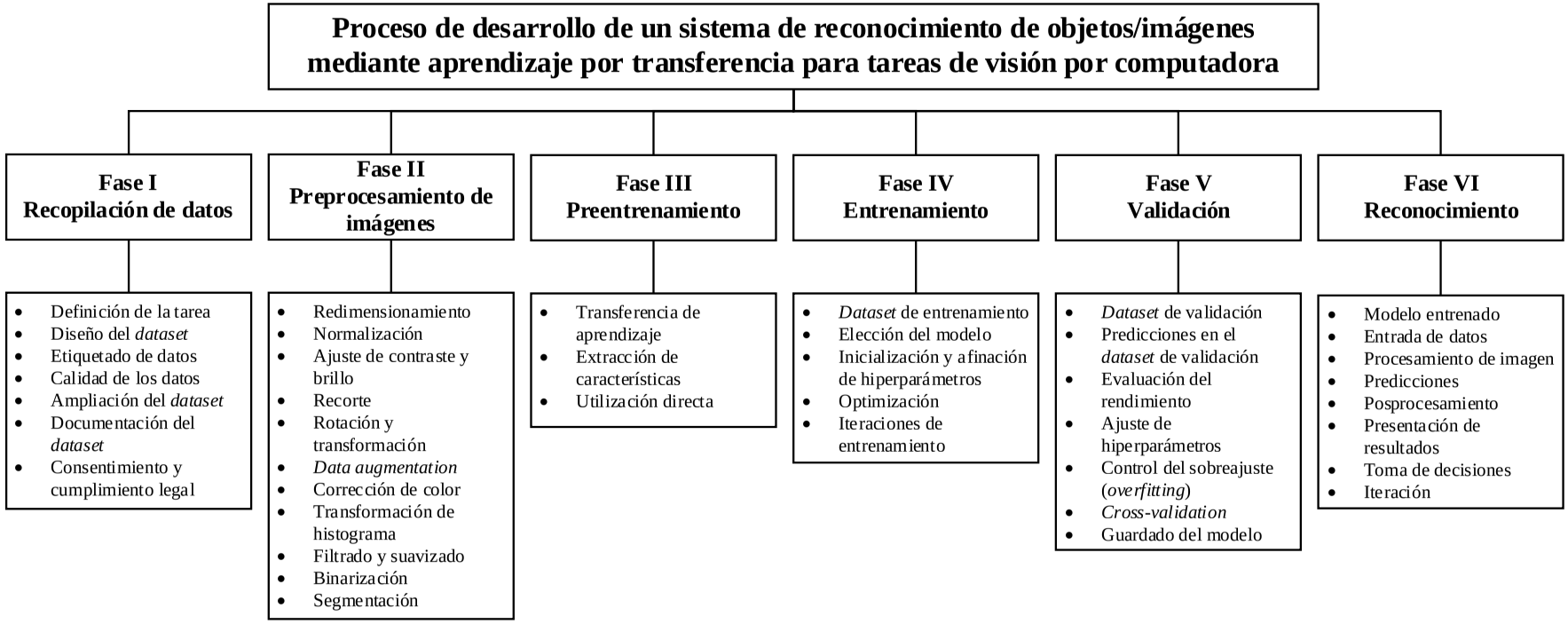
En este objetivo específico de estudio, la variable independiente es el conjunto de modelos de redes neuronales convolucionales entrenados bajo las mismas condiciones y métricas, mientras que la variable dependiente es el desempeño de estos modelos en la tarea de reconocimiento de la simbología base de los quipus de escritura, medido en términos de exactitud. Para el análisis, se emplearon los siguientes materiales: un *smartphone Redmi Note 11S* con una cámara de 108 *MP*, el entorno de desarrollo integrado Anaconda, *Python* versión 3.12.4, y una laptop con un procesador *Intel® Core™ i7-8550U* y tarjeta gráfica *NVIDIA GeForce 930 MX*. Se empleó el aprendizaje automático como técnica principal para la prueba estadística inferencial, seguido de una comparación de los resultados obtenidos para los cinco modelos. Se evaluaron y compararon funciones de pérdida, exactitudes, tiempos de entrenamiento con *GPU* y los pesos o tamaños de los modelos.

3.6 Método propuesto

En base a la existencia de métodos para el reconocimiento de escritura en diferentes idiomas (Altwaijry y Al-Turaiki, 2021; Bannigidad y Gudada, 2019; Nanehkaran et al., 2021; Neto et al., 2020; Rehman, 2021; Shams et al., 2020; Wu et al., 2017; Zhuang et al., 2021); en esta sección presentamos el método propuesto para el proceso de desarrollo de un sistema de reconocimiento de objetos o imágenes mediante aprendizaje por transferencia para tareas de visión por computadora, que consta de las siguientes fases: recopilación de datos, preprocesamiento de imágenes, preentrenamiento del modelo, entrenamiento, validación y reconocimiento.

Figura 12

Método propuesto



CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

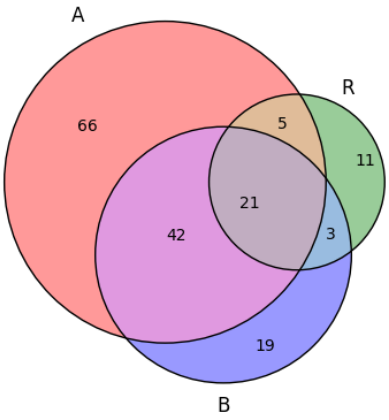
4.1 Resultados

4.1.1 Resultados conforme al objetivo específico 1

A. Análisis estadístico y matemático sobre los quipus de escritura

Figura 13

Diagrama de conjuntos de los símbolos utilizados en los quipus de escritura

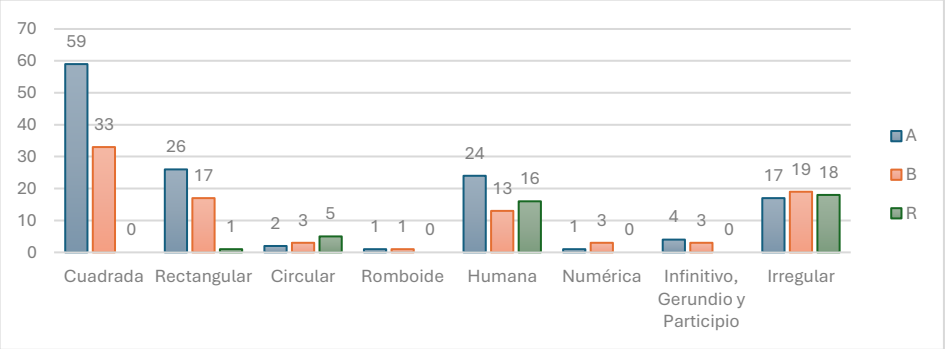


De acuerdo al Anexo 7, se construyó el diagrama de conjuntos respecto a los símbolos utilizados en los quipus de escritura por los autores A (Huillca-Huallpa, 2016), B de *Exsul Immeritus Blas Valera populo Suo* (1616) del P. Blas Valera e *Historia et Rudimenta Linguae Piruanorum*, (final del siglo XVI-1638) de los italianos H. Antonio Cumis y P. Anello Oliva (Laurencich-Minelli 2016b), y otros; y R (di Sangro, 1750); encontrándose que los 3 autores utilizan 21 símbolos que coinciden en las palabras base y que representa el 12.57% del total de símbolos identificados, los autores A y B coinciden en 63 símbolos que representa el 37.76%, los autores A y R coinciden en 26 símbolos que corresponde al 15.56% del total de símbolos identificados, los autores B y R coinciden en 24 símbolos que representan el 14.37% del total de símbolos identificados; para A no coincide con los demás en 66 que representa el 39.52%,

símbolos, B no coincide con los demás autores en 19 que representa el 11.38% y R en 11 correspondiente al 6.59%, véase en la Figura 13.

Figura 14

Formas de los símbolos base



En la Figura 14 se muestra las formas de los símbolos base de los quipus de escritura, las formas cuadradas son las más utilizadas por el autor A de 134 símbolos 59 son cuadrados que representan el 44.03% y por B de 92 símbolos 33 son cuadrados que representan el 35.87%; las formas rectangulares utilizados por A son 26 que representan el 19.40%, por B son 17 que representan el 18.48%, y por R es 1 que representan el 2.50%; las formas humanas utilizadas por A son 24 que representan el 17.91%, por B son 13 que representan el 14.13%, y por R son 16 que representan el 40.00%; las forma irregulares (cruz, máscara, arco, flores, triángulo, rayo, estrella, entre otros) utilizados por A son 17 que representan el 12.69%, por B son 19 que representan el 20.65%, y por R son 18 que representan el 45.00%; las formas circulares utilizados por A son 2 que representa el 1.49%, por B son 3 que representa el 3.26%, y por R son 5 que representa el 12.5% ; las formas infinitivo, gerundio y participio (*y*, *n*, *q*, *p*, *an*, *paq*) utilizados por A son 4 que representan el 2.99%, por B son 3 que representan el 3.26%; las formas numéricas utilizados por A es 1 que representa el 0.75%, por B son 3 que representan el 3.26%; y finalmente, las formas romboides utilizados por A es 1 que representa el 0.75%, por B es 1 que representa el 1.09%.

B. Patrones significativos más allá de la representación numérica

- Los quipus de escritura a diferencia de los quipus numéricos tienen adicionalmente símbolos que son una representación abstracta de formas (cuadrada, rectangular, circular, romboide, humana, numérica, infinitivo, gerundio y participio, e irregular), un claro ejemplo es el quipu de Andrzej Benesz (Lang, 2001), además son escritas mediante tejidos en las cuerdas colgantes, y dibujados en tablillas (Huillca-Huallpa, 2016) y tablones. Su simbología está constituida por más de 167 formas correspondientes a palabras o frases. Existen también quipus de escritura que no contienen nudos y se diferencian por el color del hilo o hilos entrelazados en las cuerdas hijas que representan silabas, un claro ejemplo son los quipus de San Juan de Collata (Hyland y Hyland, 2020).

C. Patrones recurrentes en materiales y colores empleados en las formas

- Formas elaboradas con metales (oro, plata, cobre) que son láminas en miniatura, grabados con dibujos que utilizan diferentes colores.
- Formas elaboradas en madera (tablillas y tablones), específicamente dibujados con diferentes tonalidades.
- Formas elaboradas con tejidos (trenzado con entrelazado de hilos) específicamente en la parte superior y en el contorno de la cuerda secundaria o hija.
- Cabe mencionar también que los *tocapu* (tejidos en uncus, tapicería en cumbe, dibujos grabados en aríbalos o *urpu*, keros, entre otros objetos) representan a palabras o frases.

D. Patrones recurrentes en materiales y colores empleados en las cuerdas

- Una cuerda principal trenzada con hilos de uno o varios colores.
- Una o más cuerdas colgantes de algodón o fibra de camélidos (huanaco, llama, alpaca, vicuña) de colores naturales (con

tonalidades de color blanco, crema, marrón, marrón oscuro, marrón dorado, gris, negro, entre otros) y colores artificiales (con tonalidades de color rojo, verde, azul, dorado, amarillo, magenta, cian, anaranjado, rosado, morado, entre otros).

E. Patrones recurrentes en la disposición de los nudos

- Si la cuerda colgante cuenta con una forma y no cuenta con nudos, representa una palabra o frase.
- Si la cuerda colgante cuenta con una forma y uno o más nudos en las cuerdas colgantes representan sílabas; si la cuerda tiene un nudo corresponde a la primera silaba del significado del símbolo, si la cuerda tiene dos nudos corresponde a la segunda silaba del significado del símbolo, así sucesivamente.
- No pueden existir más nudos respecto a la cantidad de sílabas del significado del símbolo base.

F. Patrones recurrentes en la codificación

- El cambio de un solo color en la forma, cambia el significado inicial de la palabra o frase.
- Los nudos codifican (cifran la palabra o frase, convirtiéndolos a sílabas) la escritura.

G. Asociaciones entre estos patrones

Tabla 1

Cantidad de palabras, sílabas y promedio de nudos en las cuerdas

Descripción	Sin repeticiones				Con repeticiones			
	A	B	R	ABR	A	B	R	ABR
Palabras/frases (símbolos)	134	85	40	167	134	92	40	266
Sílabas/fonemas	140	116	69	198	412	257	109	778
Palabras + sílabas	274	201	109	365	546	349	149	1044
Nudos por cuerda (promedio)	3.075	2.776	2.725	2.859	3.075	2.793	2.725	2.864

En la Tabla 1 se muestra la cantidad de palabras o frases (símbolos) que se pueden generar, la cantidad de sílabas que se pueden generar y el promedio de nudos en las cuerdas para A, B, R y la unión de ABR. Sin repeticiones (la silaba está en una sola palabra/frase base) se tiene 134 palabras para A, 85 palabras para B y 40 palabras para R y un total de 167 palabras sin repeticiones; además de, 140 sílabas para A, 116 sílabas para B y 69 sílabas para R y un total de 198 sílabas sin repeticiones; al sumar las palabras/frases sin repeticiones y las sílabas sin repeticiones se tiene 274 palabras y sílabas para A, 201 palabras y sílabas para B y 109 palabras y sílabas para R y un total de 365 palabras sin repeticiones y sílabas sin repeticiones; respecto a la cantidad promedio de nudos por cuerda hija se tiene 3.705 nudos por cuerda para A, 2.776 nudos por cuerda para B, 2.725 nudos por cuerda para R y un total de 2.859 nudos por cuerda hija en promedio. Con repeticiones (la silaba está en más de una palabra/frase base) se tiene 134 palabras para A, 92 palabras para B y 40 palabras para R y un total de 266 palabras con repeticiones; además de, 412 sílabas para A, 257 sílabas para B y 109 sílabas para R y un total de 778 sílabas con repeticiones; al sumar las palabras/frases con repeticiones y las sílabas con repeticiones se tiene 546 palabras y sílabas para A, 349 palabras y sílabas para B y 149 palabras y sílabas para R y un total de 1044 palabras con repeticiones y sílabas con repeticiones; respecto a la cantidad promedio de

nudos por cuerda hija se tiene 3.705 nudos por cuerda para A, 2.793 nudos por cuerda para B, 2.725 nudos por cuerda para R y un total de 2.864 nudos por cuerda hija en promedio. sílabas generadas con repetición y sin repetición se muestran en el Anexo 8.

4.1.2 Resultados conforme al objetivo específico 2

En esta parte se aplican las fases del método propuesto "Proceso de desarrollo de un sistema de reconocimiento de objetos/imágenes mediante aprendizaje por transferencia para tareas de visión por computadora", véase la Figura 12.

A. Recopilación de datos

En esa fase del estudio, se construye el conjunto de datos de imágenes con 40 clases, véase la Tabla 2, que contienen palabras base de la escritura ; como se mencionó previamente, no existe un conjunto de datos de características que se adecúen a las necesidades de la investigación en curso (al menos ninguno que conozca al día de hoy). Por este motivo, se recopiló un conjunto de datos propio que al momento consta de 4000 imágenes en crudo, de los cuales el 80% conformó el grupo de entrenamiento es decir 3200 imágenes y el 20% de validación es decir 800 imágenes; todos ellos extraídos de quipus de escritura denominados *khapaq khipu*, tablillas de madera (Huillca-Huallpa, 2016), documentos jesuíticos (Laurencich-Minelli, 2016b), manuscritos (Animato et al., 1989) y crónicas (Zavala et al., 2021). Debido a que se tuvo un conjunto de datos reducido, se generaron datos sintéticos al igual que (Sutramiani et al., 2021), es decir que en algunos casos se crearon imágenes de manera manual.

Tabla 2

40 clases o palabras base de la escritura Inca

N°	*Class o primary words*	*Meaning*
1	*PACHACAMAC*	*God, creator of universe*
2	*VIRACOCHA*	*God, human form*
3	*YNTI*	*Sun*
4	*QUILLA*	*Moon*
5	*CHASCA*	*Venus*
6	*COYLLUR*	*Star*
7	*HIPUY*	*Comet*
8	*CUYCHU*	*Rainbow*
9	*YLLAPA*	*Ray*
10	*PINUNSUN*	*Equinox*
11	*MAYTIÑU*	*Sun eclipse*
12	*YANRIÑUY*	*Moon eclipse*
13	*YNCA*	*King*
14	*COYA*	*Queen*
15	*AUQUI*	*Prince*
16	*ÑUSTA*	*Princess*
17	*MANCO CAPAC*	*First Inca*
18	*OCLLO*	*First Queen*
19	*SINCHI ROCA*	*Second Inca*
20	*MAMA CORA*	*Second Queen*
21	*CURACA*	*Feudatory, noble*
22	*RUNA*	*Man*
23	*HANAN PACHA*	*Sky*
24	*UCU PACHA*	*Underground*
25	*TUTA*	*Night*
26	*PUMA*	*Cougar*
27	*UCUMARI*	*Spectacled bear*
28	*UTURUNCU*	*Jaguar, otorongo*
29	*SURI*	*Rhea*
30	*CUNTUR*	*Condor*
31	*URITU*	*Parrot*
32	*UNUY*	*Water*
33	*LLAUTU*	*Colored braid*
34	*AMARU*	*Snake*
35	*CITU*	*Sun honoring*
36	*MUNCAYNIM*	*Pan flaute*
37	*CATOLLAY*	*Duel*
38	*QUINQUIR*	*Student dress*
39	*CANTUT*	*Cantuta flower*
40	*TACVEHIRAC*	*Sling*

Nota. Esta tabla muestra las 40 clases o palabras base de la escritura Inca. Tomado de *Quipus, computation and generative grammars* (p. 359), por Roncoroni, (2017), *GA2018 – XXI Generative Art Conference*.

Se asignaron etiquetas del vocablo quechua a cada carpeta perteneciente a cada clase, como se ilustra en la Figura 15, tanto en el grupo de entrenamiento como en el grupo de validación, además, se asignaron etiquetas a cada imagen en el conjunto de datos. Para asegurar la calidad se eliminó las imágenes defectuosas.

Figura 15

Etiquetas para las 40 clases

El conjunto de datos construido corresponde aproximadamente al 23.95% (40 clases) del total de 167 clases, tal como se muestra en la Tabla 3, cada clase constó de 4000 imágenes distribuidas uniformemente, con 100 imágenes por clase. Cada imagen con una resolución de 150×385 píxeles y en formato JPG, lo que facilitó su manejo y análisis.

Tabla 3

Una breve descripción del conjunto de datos

Descripción	Valor
Número total de clases	40
Número de instancias en cada clase	100
Número total de muestras	$40 \times 100 = 4000$
Tamaño de cada imagen	150×385
Formato de cada imagen	JPG

B. Preprocesamiento de imágenes

En esta segunda fase, se redimensionó el tamaño de las imágenes para que tengan las mismas dimensiones (224×224 *píxeles* o 299×299 *píxeles*) y se convirtió las imágenes a nivel de escala de grises, además, se utilizó la rotación, traslación, transformaciones afines y segmentación; y todos los archivos en formato *.jpg*. Los símbolos pictográficos presentan ruido y su disponibilidad de datos es limitada, con la finalidad de mejorar el rendimiento del modelo de reconocimiento se procedió a realizar recortes, ajustes de contraste y brillo, corrección de color, eliminación de artefactos, filtrado y suavizado.

C. Preentrenamiento

En esta fase los modelos preentrenados son modelos que han sido entrenados previamente en el conjunto de datos *ILSVR* para resolver tareas específicas, las arquitecturas de red preentrenadas se muestran en la Tabla 4. Estos modelos tienen la capacidad impresionante para reconocer imágenes.

Tabla 4

Resumen de las arquitecturas de red preentrenadas

Arquitectura	Profundidad	Tamaño de entrada
AlexNet	8	$224 \times 224 \times 3$
VGG11-BN	11	$224 \times 224 \times 3$
SqueezeNet	18	$224 \times 224 \times 3$
DenseNet121	121	$224 \times 224 \times 3$
Inceptionv3	48	$299 \times 299 \times 3$

D. Entrenamiento

En la fase de entrenamiento, se preparó el conjunto de datos de entrenamiento de la red, se eligieron para dicha tarea cinco modelos *DCNN* preentrenados (*AlexNet*, *VGG11-BN*, *SqueezeNet*, *DenseNet121*, e *Inceptionv3*) en *PyTorch*. Aquí se muestra una breve descripción de los modelos utilizados:

AlexNet es una *CNN* con imágenes *RGB* de 224×224 pixeles de entrada, 5 capas de convolución, 3 capas de agrupación máxima, utiliza 7 capas *ReLU* como función de activación, una capa de agrupación promedio adaptable, 2 capas *Dropout*, 3 capas de rectificación lineal y utiliza *Softmax* como función de activación en la capa de salida (Krizhevsky et al., 2017).

VGG11-BN es una *CNN* con imágenes *RGB* de 224×224 pixeles de entrada, 8 capas de convolución, 8 capas normalización de lotes, 5 capas de agrupación máxima, utiliza 10 capas *ReLU* como función de activación, una capa de agrupación promedio adaptable, 2 capas *Dropout*, 3 capas de rectificación lineal y utiliza *Softmax* como función de activación en la capa de salida (Simonyan y Zisserman, 2015).

SqueezeNet es una *CNN* con imágenes *RGB* de 224×224 pixeles de entrada, 2 capas de convolución, 3 capas de agrupación máxima, 8 capas *Fire*, utiliza 2 capas *ReLU* como función de activación, una capa de agrupación promedio adaptable, una capa *Dropout* y utiliza *Softmax* como función de activación en la capa de salida (Iandola et al., 2016).

DenseNet121 es una *CNN* con imágenes *RGB* de 224×224 pixeles de entrada, una capa de convolución, 2 capas de normalización de lotes, una capa de agrupación máxima, 4 capas *DenseBlok* que incluyen en total 58 capas *DenseLayer*, 3 capas de transición, utiliza una capa *ReLU* como función de activación, una capa de rectificación lineal y utiliza *Softmax* como función de activación en la capa de salida (Huang et al., 2017).

Inceptionv3 es una *CNN* con imágenes *RGB* de 299×299 pixeles de entrada, 90 capas de convolución básica, 2 capas de agrupación máxima, una capa de agrupación promedio adaptable, una capa *Dropout*, 2 capas de rectificación lineal, 3 capas *InceptionA*, una capa *InceptionB*, 4 capas *InceptionC*, una capa *InceptionAux*, una capa *InceptionD*, 2 capas *InceptionE* (Szegedy et al., 2016).

Tabla 5

Valores de hiperparámetros

Parámetro	Valor
Tipo de entorno de ejecución	*GPU*
Tamaño del vector de entrada	224 o 299 (*Inceptionv3*)
Clases	40
Optimizador	*SGD*
Función de activación	*ReLU*
Tasa de aprendizaje	0.001
Momentum	0.9
Dropout probability	0.5
Batchsize	8
Número de épocas	100

Se desarrolló en un entorno *Python*, se utilizaron varias bibliotecas de código abierto para la tarea: *opencv*, *torch*, *torchvision*, *matplotlib*, *numpy*, *datetime*, *pathlib*, *pillow*, *skimage*, entre otros. Respecto a las hiperparámetros, véase la Tabla 5, los modelos se entrenan con el optimizador *SGD* con una tasa de aprendizaje de 0.001 y un valor de *momentum* de 0.9, tiene como imágenes de entrada de 224×224 píxeles para *AlexNet*, *VGG11-BN*, *SqueezeNet* y *DenseNet121*, mientras que para *Inceptionv3* las imágenes de entrada de 299×299 píxeles, el número de clases es 40, el número de épocas es 100, el tamaño del lote 8, Los experimentos se simularon en la potente plataforma *Google Colaboratory*. El laboratorio contó con una tarjeta de video *NVIDIA Tesla K*80, *2496 CUDA cores*, *12 GB GDDR5-VRAM GPU*. Durante el proceso de entrenamiento, los pesos finales se guardan en un modelo que es capaz de detectar la simbología base de los quipus de escritura, y es capaz de establecer a que clases pertenece. El código fuente para el entrenamiento se muestra en el Anexo 2.

E. Validación

En esta fase, se prepara el conjunto de datos de validación independiente, que corresponden a imágenes no utilizados en

entrenamiento, el modelo entrenado (*AlexNet*, *VGG11-BN*, *SqueezeNet*, *DenseNet121*, e *Inceptionv3*) se utiliza para realizar predicciones. Se compara la salida del modelo con las etiquetas reales en el conjunto de validación, y el rendimiento del modelo se evaluó mediante métricas función de pérdida y exactitud. Además, se midió el tiempo de entrenamiento del modelo y se establece la mejor exactitud con los datos de validación. El código fuente para generar la matriz de confusión a partir del mejor modelo, se muestra en el Anexo 3, y la matriz de confusión se ilustra en el Anexo 6.

F. Reconocimiento

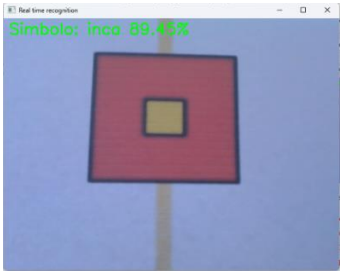
En esta fase se utiliza el modelo previamente entrenado *best.pt* para identificar las imágenes. Se proporciona la ruta de una imagen digital cargada desde un archivo, luego se realiza el preprocesado de la imagen a reconocer; y finalmente la presentación de resultados consiste en mostrar la etiqueta de la clase a la cual pertenece la imagen y una probabilidad asociada a la clase, el código fuente para la predicción o reconocimiento, se muestra en el Anexo 4.

F.1 Reconocimiento en tiempo real

Para probar aún más el rendimiento de este modelo, se organiza un reconocimiento en tiempo real. Esta prueba se realiza en base al modelo *DenseNet121* que se entrenó en 100 épocas, se probaron con 40 símbolos básicos en tiempo real (véase Figura 16), para usarlos como muestras de prueba, una muestra de cada clase.

Figura 16

Reconocimiento en tiempo real



El resultado indica que se han identificado adecuadamente 19 muestras, aunque muchas de estas palabras base están escritas en un fondo complejo y los símbolos están compuestos de muchos colores, esto causa una enorme dificultad para el modelo entrenado como indica (Zhuang et al., 2021). El código fuente para el reconocimiento en tiempo real se muestra en el Anexo 5.

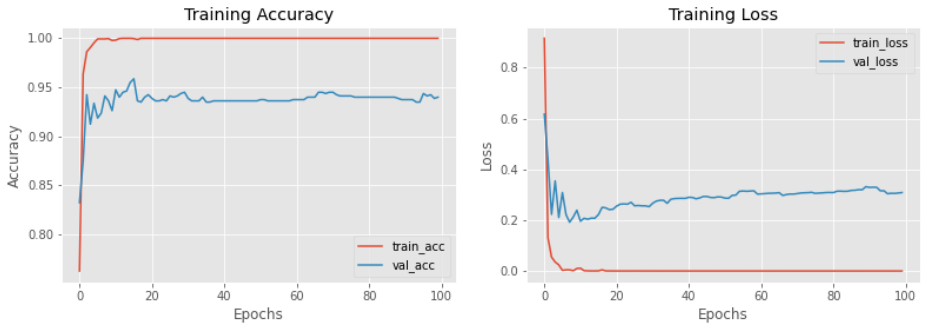
4.1.3 Resultados conforme al objetivo específico 3

A. Exactitud y función de pérdida

Bajo las mismas condiciones y métricas, se entrenaron cinco modelos *DCNN* utilizando aprendizaje por transferencia, con el conjunto de datos propio correspondiente a imágenes de la simbología base de los quipus de escritura, que constó de 4000 imágenes en total correspondientes a 40 clases; de los cuales 3200 imágenes se utilizaron en la fase de entrenamiento y 800 imágenes en la fase de prueba. Con base en los valores de hiperparámetros enumerados en la Tabla 5, en este experimento entrenamos y validamos las cinco arquitecturas preentrenadas para la simbología base utilizando nuestro conjunto de datos.

Figura 17

Métricas exactitud y función de pérdida al entrenar AlexNet

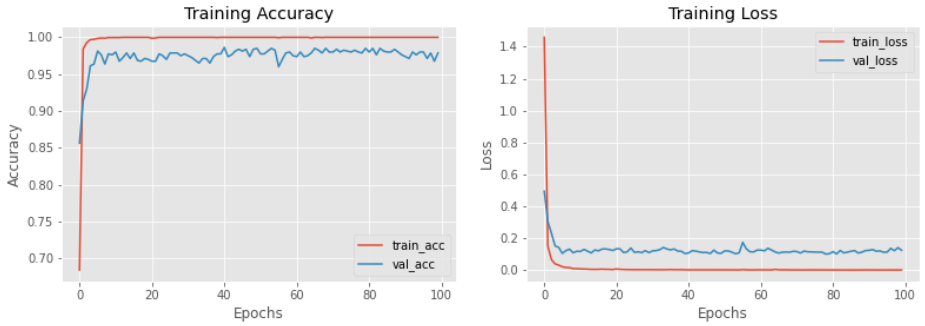


La Figura 17 muestra los resultados del modelo *AlexNet* utilizando aprendizaje por transferencia. Durante la fase de validación, la exactitud del modelo varió desde un mínimo de 83.25% hasta un máximo de 95.87%, indicando una notable variabilidad en el rendimiento. Además, la

función de pérdida más baja registrada en la fase de validación fue de 0.1918, lo que refleja el mejor ajuste obtenido por el modelo en esta fase.

Figura 18

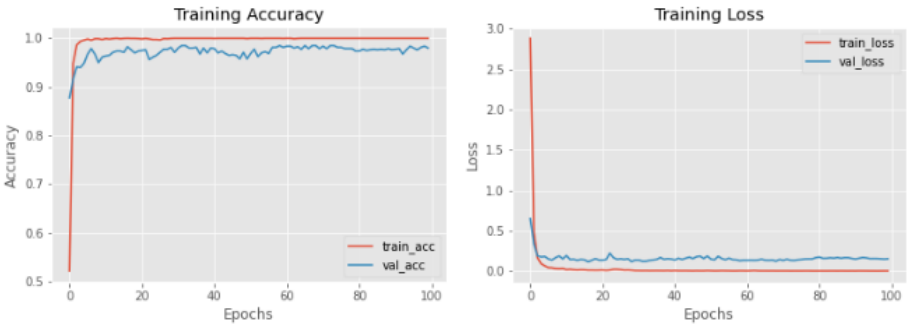
Métricas exactitud y función de pérdida al entrenar DenseNet121



La Figura 18 presenta los resultados del modelo *DenseNet121* utilizando aprendizaje por transferencia. En la fase de validación, la exactitud del modelo varió entre un mínimo de 85.63% y un máximo de 98.63%, mostrando un notable potencial para alcanzar alta precisión en ciertas condiciones. Además, la menor función de pérdida registrada fue de 0.1003, indicando el mejor ajuste del modelo a los datos de validación y sugiriendo una capacidad destacada para minimizar el error en la clasificación.

Figura 19

Métricas exactitud y función de pérdida al entrenar Inceptionv3

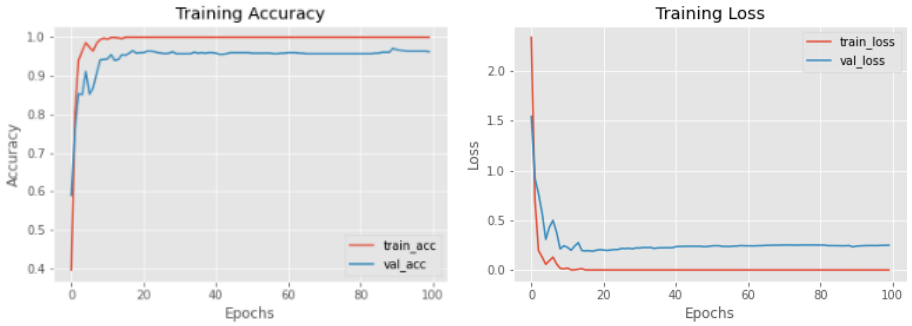


La Figura 19 presenta los resultados del modelo *Inceptionv3* con aprendizaje por transferencia. Durante la fase de validación, la exactitud del modelo varió entre un mínimo de 87.75% y un máximo de 98.50%, reflejando una alta precisión en varios casos. La función de pérdida más baja registrada fue de 0.1154, lo que demuestra un buen ajuste del modelo

a los datos de validación, con una capacidad notable para reducir el error en la clasificación.

Figura 20

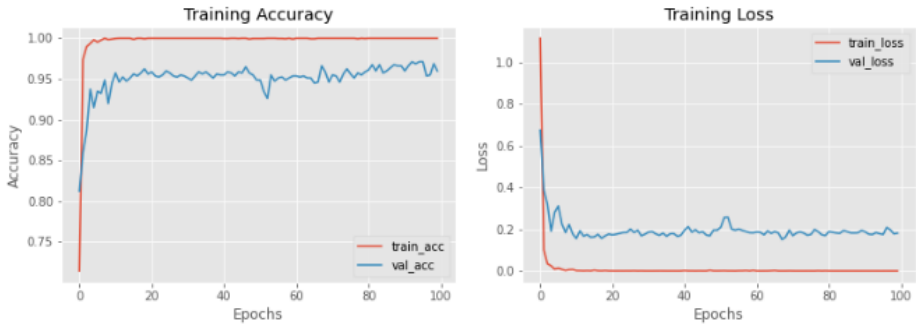
Métricas exactitud y función de pérdida al entrenar SqueezeNet



La Figura 20 muestra los resultados del modelo *SqueezeNet* con aprendizaje por transferencia durante 100 épocas. La exactitud en la fase de validación varió entre 59.00% y 97.13%, indicando una considerable variabilidad en el rendimiento. La menor función de pérdida registrada fue de 0.1884, lo que muestra un buen ajuste del modelo a los datos de validación en el escenario óptimo.

Figura 21

Métricas exactitud y función de pérdida al entrenar VGG11-BN



La Figura 21 muestra los resultados del modelo *VGG11-BN* con aprendizaje por transferencia. Durante la fase de validación, la exactitud del modelo osciló entre un mínimo de 81.25% y un máximo de 97.13%, lo que indica un desempeño variable, pero con un alto potencial. La función de pérdida más baja observada fue de 0.1510, sugiriendo un buen ajuste del modelo a los datos de validación en términos de minimizar el error.

Tabla 6

Función de pérdida, exactitud, tiempo de entrenamiento y peso

Modelos	Validación		Tiempo de entrenamiento en 100 épocas (*GPU*)	Peso o tamaño del modelo (*KB*)
	Función de pérdida	Exactitud		
AlexNet	0.1918	95.87%	75m 43s	446633
VGG11-BN	0.1510	97.13%	177m 34s	1007363
SqueezeNet	0.1884	97.13%	42m 58s	5940
DenseNet121	0.1003	98.63%	109m 24s	55344
Inceptionv3	0.1154	98.50%	146m 32s	191485

La Tabla 6 resume los resultados obtenidos al entrenar cinco modelos durante 100 épocas, incluyendo el tiempo de entrenamiento utilizando *GPU* y el tamaño del archivo generado. Entre los modelos evaluados, *DenseNet121* destaca como el mejor en términos de rendimiento, alcanzando una exactitud máxima de 98.63% y la menor función de pérdida de 0.1003, con un tiempo de entrenamiento de 109 minutos y 24 segundos. Además, el modelo *SqueezeNet* demostró ser el más eficiente en tiempo de entrenamiento, con solo 42 minutos y 58 segundos, y también es el de menor tamaño, con 5940 *KB*. Estos resultados sugieren que, aunque *DenseNet121* ofrece el mejor desempeño en precisión y ajuste, *SqueezeNet* puede ser preferible en situaciones donde el tiempo y el tamaño del modelo son críticos.

4.2 Discusión

Respecto a las palabras primarias, en nuestro estudio hallamos 69 sílabas con la data de palabras/frases (simbología) de di Sangro (1750); Roncoroni (2017) en la misma data encuentra un total de 76 sílabas.

En nuestro estudio hemos construido un conjunto de datos de imágenes correspondientes a las palabras/frases (simbología) base de la escritura en quipus y en otras investigaciones se utilizan conjunto de datos disponibles en la web, como se muestra en la Tabla 7, además, nuestro conjunto de datos construido que consta de 4000 imágenes pertenecientes a 40 clases, existen estudios con 10 clases (Agrawal et al., 2021; Agrawal

y Awasthi, 2021; Elleuch et al., 2021; Nanehkaran et al., 2021; Neto et al., 2020; Pratama, 2021; Rajpal y Garg, 2023b; Riaz et al., 2022) hasta con 7356 clases (Wu et al., 2017); al aplicar el aumento de datos, el modelo logró mejores resultados como en el estudio de AlJarrah et al. (2021). Respecto al tipo y escritura, existen estudios en caracteres, dígitos, y palabras en las escrituras: Devanagari, China, Coreana, Kannada, Meitei Mayek, Arábiga, Balinese, Bangla, Urdu, Bengali e Hindi; en nuestro estudio utilizamos el tipo de escritura palabra en la escritura Inca.

Respecto a la métrica exactitud en la validación existen estudios como se muestra en la Tabla 7, en donde podemos ver que en la investigación de Chandrakala y Thippeswamy (2020) obtienen el 70.00% de exactitud en la validación con el modelo *AlexNet*, mientras que en nuestro estudio obtuvimos 95.87%, como se ilustra en la Figura 22; además, utilizando *VGG11-BN* obtuvimos una mejor exactitud de 97.13%, y con *SqueezeNet* de 97.13%; existen tres estudios con el modelos *DenseNet121* los estudios de Ghosh et al., (2021); Riaz et al., (2022); y de Sandhya y Geetha (2022) que obtienen exactitudes en la validación de 96.55%, 94.00% y 96.28% respectivamente, en tanto nosotros obtenemos el 98.63%, siendo este el mejor resultado que alcanzamos, esto podría deberse a uso de modelos preentrenados en este estudio ya que contamos con un conjunto de datos muy pequeño; también existen tres estudios con el modelo *Inceptionv3* los estudios de Ghosh et al. (2021); Purnamawati et al. (2018b); y Riaz et al. (2022) que obtienen exactitudes en la validación de 86.90%, 96.20% y 95.00% respectivamente, resultados similares a nosotros que obtenemos el 98.50% con el mismo modelo. El entrenamiento para el reconocimiento de escritura lo realizan en 4 épocas como los estudios de Sridevi y Rangarajan (2021) a 5000 épocas como en el estudio de Zhuang et al. (2021), y nosotros entrenamos nuestros modelos en 100 épocas. Respecto al tiempo promedio de entrenamiento por época en el estudio de Deore y Pravin (2020) el resultado es de 3.63m y de 2.58m en el entorno de ejecución *NVIDIA Tesla K*80 *GPU*, en nuestra investigación el tiempo promedio de entrenamiento por época para el modelo *DenseNet121* fue de 1.09m en el entorno de ejecución *NVIDIA Tesla K*80, *2496 CUDA cores*, *12 GB GDDR5-VRAM GPU*.

Figura 22

Métrica exactitud de los cinco modelos en la fase de validación

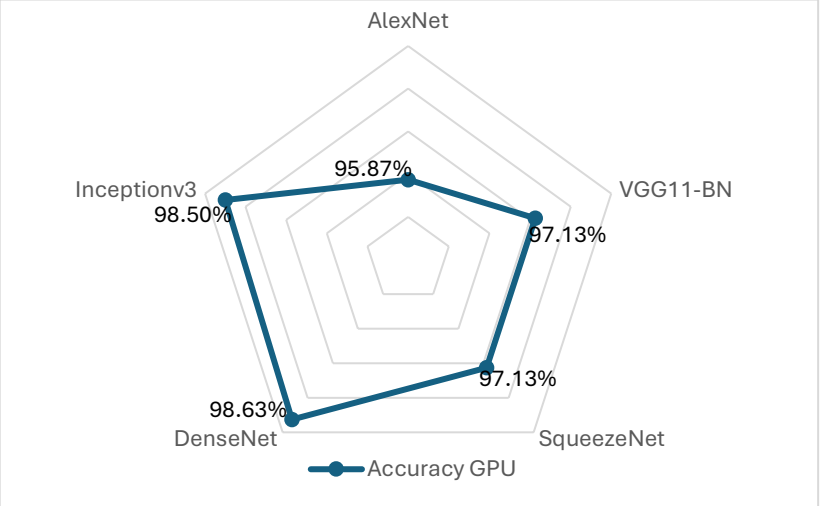


Tabla 7

Exactitud en diferentes estudios sobre reconocimiento de escritura

N°	Referencia	Conjunto de datos	Clases	Tipo	Escritura	Modelo	Épocas	Exactitud *Train*	Exactitud *Val*	TxE[1]	Entorno de ejecución
1	(Acharya et al., 2015)	*DHCD*	46	Caracter	Devanagari		50		98.47%		
2	(Zhong et al., 2015)	*SCUT-SPCCI*	3755	Caracter	China			99.74%	94.38%		
3	(Wu et al., 2017)	*CASIA-HWDB*	7356	Caracter	China		90		95.88%		*NVIDIA Titan X GPU*
4	(Wu et al., 2017)	*ICDAR-2013*		Caracter	China		90		96.20%		*NVIDIA Titan X GPU*
5	(Purnamawati et al., 2018a)		11	Caracter	Coreana	*Inceptionv3*	4000		86.90%		
6	(Aneja y Aneja, 2019)	*DHCD*		Caracter	Devanagari	*Inceptionv3*	15	99.00%		16.3m	
7	(Aneja y Aneja, 2019)	*DHCD*		Caracter	Devanagari	*AlexNet*	15	98.00%		2.2m	
8	(Chandrakala y Thippeswamy, 2020)		118	Caracter	Kannada	*AlexNet*	10		70.00%		
9	(Deore y Pravin, 2020)	*UCI*	58	Caracter, Digito	Devanagari	*VGG16*	20	95.70%	94.83%	3.63m	*NVIDIA Tesla K80 GPU*
10	(Deore y Pravin, 2020)	*UCI*	58	Caracter, Digito	Devanagari		10		96.55%	2.58m	*NVIDIA Tesla K80 GPU*
11	(Gurav et al., 2020)		30	Caracter	Devanagari		15	99.65%	98.22%		
12	(Inunganbi et al., 2021)		35	Caracter	Meitei Mayek		10	98.70%			
13	(Neto et al., 2020)	*CAR-A, CAR-B, CVL HDS, BBC-CAR*	10	Digito	Arábiga		20	93.54%			*NVIDIA Tesla P100 GPU*

[1] Tiempo promedio de entrenamiento por época

N°	Referencia	Conjunto de datos	Clases	Tipo	Escritura	Modelo	Épocas	Exactitud *Train*	Exactitud *Val*	TxE[2]	Entorno de ejecución
14	(Rani et al., 2020)		188	Caracter	Kannada		10	73.51%			
15	(Shams et al., 2020)	*AHCD*	28	Caracter	Arábiga		400	98.08%	95.07%		*NVIDIA 4G-GT 740m GPU*
16	(Agrawal y Awasthi, 2021)	*ARDIS*	10	Digito	Arábiga			99.76%	98.70%		
17	(Agrawal y Awasthi, 2021)	*USPS*	10	Digito	Arábiga			98.22%	93.01%		
18	(Agrawal et al., 2021)	*MNIST*	10	Digito	Arábiga		10	99.06%	98.80%		
19	(AlJarrah et al., 2021)	*AHCD*	28	Caracter	Arábiga		50	97.70%			
20	(Altwaijry y Al-Turaiki, 2021)	*AHCD*	28	Caracter	Arábiga		30		93.84%		*NVIDIA RTX 2080 Ti GPU*
21	(Altwaijry y Al-Turaiki, 2021)	*Hijja*	29	Caracter	Arábiga		30		80.00%		*NVIDIA RTX 2080 Ti GPU*
22	(Elleuch et al., 2021)	*IFN/ENIT*	10	Palabra	Arábiga	*ResNet*	10	98.99%			
23	(Elleuch et al., 2021)	*IFN/ENIT*	10	Palabra	Arábiga	*VGG16*	10	98.10%			
24	(Elleuch et al., 2021)	*IFN/ENIT*	10	Palabra	Arábiga	*Inceptionv3*	10	95.70%			
25	(Nanehkaran et al., 2021)	*HODA*	10	Digito	Farsi		5	99.91%	99.45%		
26	(Pratama, 2021)	*MNIST*	10	Digito	Arábiga	*LeNet-5*	30		99.03%		
27	(Rajpal et al., 2021)	*DHCD*	35	Caracter	Devanagari		60		98.73%		*NVIDIA Tesla K80 GPU*
28	(Sridevi y Rangarajan, 2021)		156	Caracter	Kannada		4	99.51%			
29	(Sutramiani et al., 2021)	*HBCL*	18	Caracter	Balinese	*VGG19*	150	99.82%	96.29%		
30	(Sutramiani et al., 2021)	*HBCL*	18	Caracter	Balinese	*InceptionResNetv2*	150	91.88%	82.40%		
31	(Sutramiani et al., 2021)	*HBCL*	18	Caracter	Balinese	*DenseNet169*	150	95.94%	90.74%		

[2] Tiempo promedio de entrenamiento por época

N°	Referencia	Conjunto de datos	Clases	Tipo	Escritura	Modelo	Épocas	Exactitud *Train*	Exactitud *Val*	TxE[3]	Entorno de ejecución
32	(Sutramiani et al., 2021)	*HBCL*	18	Caracter	Balinese	*ResNet152v2*	150	99.68%	95.37%		
33	(Sutramiani et al., 2021)	*HBCL*	18	Caracter	Balinese	*MobileNetv2*	150	99.61%	96.29%		
34	(Ghosh et al., 2021)	*CMATERdb*		Caracter	Bangla	*InceptionResNetv2*	50		96.99%		
35	(Ghosh et al., 2021)	*CMATERdb*		Caracter	Bangla	*DenseNet121*	50		96.55%		
36	(Ghosh et al., 2021)	*CMATERdb*		Caracter	Bangla	*Inceptionv3*	50		96.20%		
37	(Ghosh et al., 2021)	*CMATERdb*		Caracter	Bangla	*InceptionResNetv2, Inceptionv3, DenseNet121*	50		97.69%		
38	(Zhuang et al., 2021)	*CASIA-HWDB1.1*	3755	Caracter	China		5000	92.28%	90.91%		
39	(Awni et al., 2022)	*AlexU-W, IFN/ENIT*	109	Palabra	Arábiga	*ResNet18*		96.11%			*NVIDIA Tesla T4 GPU*
40	(Riaz et al., 2022)		10	Digito	Urdu	*ResNet50*	10	97.05%	96.00%		
41	(Riaz et al., 2022)		10	Digito	Urdu	*Inceptionv3*	10	96.55%	95.00%		
42	(Riaz et al., 2022)		10	Digito	Urdu	*VGGNet16*	10	96.33%	95.00%		
43	(Riaz et al., 2022)		10	Digito	Urdu	*DenseNet121*	10	95.99%	94.00%		
44	(Sandhya y Geetha, 2022)			Caracter	Kannada	*DenseNet121*		96.70%	96.28%		
45	(Sikder et al., 2022)	*BanglaLekha-Isolated*	84	Caracter	Bengalí			94.73%			
46	(Masruroh et al., 2023)	*AHCD*	28	Caracter	Arábiga	*VGG16*	100	97.64%	95.77%		
47	(Masruroh et al., 2023)	*Hijja*	29	Caracter	Arábiga	*VGG17*	100	93.59%	87.33%		
48	(Rajpal y Garg, 2023b)	*LDHCD*	10	Digito	Hindi		40		99.72%		

[3] Tiempo promedio de entrenamiento por época

CONCLUSIONES

- El quipu de escritura constituye un sistema excepcionalmente complejo y multifacético. Es simbólico, pictográfico, ideográfico, geométrico, abstracto, monocromático, policromático, logosilábico, numérico y cifrado. Está compuesto por una cuerda principal horizontal de la cual cuelgan cuerdas secundarias verticales. Estas cuerdas secundarias están unidas en la parte superior a la cuerda principal y presentan formas que representan sílabas cuando contienen uno o más nudos, o palabras y frases cuando están desprovistas de nudos. El color de las formas modifica el significado de las palabras o frases. Asimismo, existen cuerdas secundarias sin formas y que dependen de los colores de los hilos torcidos para representar palabras o frases, si tienen nudos estas cuerdas representan sílabas. El quipu también se utiliza para registros numéricos en base decimal, a veces tras realizar cálculos matemáticos realizados con la *yupana*. Respecto al material empleado en las formas, están escritas en metales, madera, trenzado con entrelazado de hilos, y *tocapu*; este último no está asociado a las cuerdas principales e hijas, y representan palabras o frases. El sistema abarca más de 167 símbolos base o palabras/frases, y más de 198 sílabas o fonemas, con un promedio de entre 2 y 3 nudos por palabra. El quipu, por tanto, no solo refleja una forma de escritura compleja, sino que también resalta la riqueza cultural y la sofisticación de la civilización andina.

- En este estudio se ha desarrollado un conjunto de datos significativo compuesto por 4,000 imágenes de quipus de escritura, abarcando 40 clases distintas, lo que representa aproximadamente el 23.95% de las 167 clases identificadas hasta el momento. Este conjunto de datos ha permitido entrenar el modelo *DenseNet121* utilizando aprendizaje por transferencia, previa formación en el conjunto de datos de *ImageNet*. Los resultados obtenidos demuestran la viabilidad y efectividad del modelo para la tarea de clasificación de quipus, tanto en el entrenamiento como en la validación del modelo, evidenciando su aplicación efectiva en el dominio específico de los quipus.

- La evaluación de cinco modelos de redes neuronales convolucionales —*AlexNet*, *VGG11-BN*, *SqueezeNet*, *DenseNet121* e *Inceptionv3*— bajo las mismas condiciones y métricas revela que *DenseNet121* es el mejor modelo en la tarea de reconocimiento de la simbología base de los quipus de escritura, alcanzando una exactitud de 98.63% en la validación. Este estudio demuestra que las redes neuronales convolucionales profundas superan a los algoritmos tradicionales de aprendizaje automático en términos de aprendizaje automático de características y resultados óptimos en el reconocimiento automático de escritura en tiempo real. El desarrollo del sistema basado en la red neuronal convolucional profunda ha logrado resultados superiores en la identificación de la simbología base, evidenciando una exactitud superior a la de otros métodos. Estos resultados no solo validan la eficacia del modelo, sino que también subrayan el potencial de las tecnologías de inteligencia artificial para mejorar la identificación y el desciframiento de textos históricos, proporcionando una herramienta robusta y eficiente para futuros estudios.

RECOMENDACIONES

- A los futuros investigadores en los quipus como sistema de escritura se recomienda el estudio de los quipus con cuerdas hijas sin las formas y con nudos, estos hilos representan sílabas, y se distinguen por el material, la cantidad de hilos entrelazados utilizados y los colores. Además, se recomienda identificar más formas o símbolos correspondientes al grupo de los infinitivos, gerundios y participios, debido a su complejidad poniendo especial énfasis en el color.

- A los que planeen utilizar el software se les recomienda que actualicen un conjunto de datos significativo de imágenes los quipus de escritura, abarcando las 167 clases, para así poder generar más de 198 sílabas/fonemas; y utilizar entre las distintas 365 palabras/frases o sílabas/fonemas. Esto asegurará una representación más completa y diversa del sistema de escritura quechua, facilitando así un análisis más profundo y preciso de sus características y variaciones.

- A los futuros investigadores en redes neuronales convolucionales profundas se les recomienda ampliar el estudio utilizando otras técnicas como *MobileNet, ResNet, GoogleNet, ShuffleNet, ZFNet-512, EfficientNet-Lite4*, entre otros; además de los modelos estudiados en esta investigación. Esto permitirá explorar y comparar diversas arquitecturas para obtener un entendimiento más completo de sus aplicaciones y rendimientos en diferentes escenarios y conjuntos de datos. Se sugiere utilizar el *framework PyTorch* para el reconocimiento automático y evaluar con métricas como exactitud, tasa de error, exhaustividad o sensibilidad, especificidad, precisión, valor predictivo negativo, valor-F, tasa de falsos positivos, tasa de falsos negativos, tasa de descubrimiento falso, tasa de omisión falsa, prevalencia, razón de verosimilitud positiva, razón de verosimilitud negativa, razón de probabilidad de diagnóstico y matriz de confusión.

BIBLIOGRAFÍA

Abderrahim, N. Y. Q., Abderrahim, S., & Rida, A. (2020). Road Segmentation using U-Net architecture. *2020 IEEE International Conference of Moroccan Geomatics (Morgeo)*, 1–4. https://doi.org/10.1109/morgeo49228.2020.9121887

Acharya, S., Pant, A. K., & Gyawali, P. K. (2015). Deep learning based large scale handwritten Devanagari character recognition. *2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, 1–6. https://doi.org/10.1109/SKIMA.2015.7400041

Agrawal, A. K., & Awasthi, V. K. (2021). Novel Deep Neural Network Model for Handwritten Digit Classification and Recognition. *Learning*, *2*(2). https://doi.org/10.48175/IJARSCT-781

Agrawal, A. K., Shrivas, A. K., & Awasthi, V. k. (2021). A Robust Model for Handwritten Digit Recognition using Machine and Deep Learning Technique. *2021 2nd International Conference for Emerging Technology (INCET)*, 1–4. https://doi.org/10.1109/INCET51464.2021.9456118

Aizenberg, I., & Butakoff, C. (2008). A windowed Gaussian notch filter for quasi-periodic noise removal. *Image and Vision Computing*, *26*(10), 1347–1353. https://doi.org/10.1016/j.imavis.2007.08.011

AlJarrah, M. N., Mo'ath, M. Z., & Duwairi, R. (2021). Arabic handwritten characters recognition using convolutional neural network. *2021 12th International Conference on Information and Communication Systems (ICICS)*, 182–188. https://doi.org/10.1109/ICICS52457.2021.9464596

Alla, S., & Adari, S. K. (2019). *Beginning anomaly detection using python-based deep learning*. Springer. https://doi.org/10.1007/978-1-4842-5177-5

Alsaeedi, A., Mutawa, H. Al, Snoussi, S., Natheer, S., Omri, K., & Subhi, W. Al. (2018). Arabic words Recognition using CNN and TNN on a Smartphone. *2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR)*, 57–61. https://doi.org/10.1109/ASAR.2018.8480267

Altwaijry, N., & Al-Turaiki, I. (2021). Arabic handwriting recognition system using

convolutional neural network. *Neural Computing and Applications*, *33*(7), 2249–2261. https://doi.org/10.1007/s00521-020-05070-8

Aneja, N., & Aneja, S. (2019). Transfer learning using CNN for handwritten devanagari character recognition. *2019 1st International Conference on Advances in Information Technology (ICAIT)*, 293–296. https://doi.org/10.1109/ICAIT47043.2019.8987286

Animato, C., Rossi, P. A., & Miccinelli, C. (1989). *Quipu, Il nodo parlante dei misteriosi Incas* (2° edizion). ECIG.

Ansari, M. S., Wasid, M., & Rahman, S. A. (2022). Devanagari Handwritten Character Recognition using Transfer Learning with Deep CNN and SVM. *2022 5th International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT)*, 1–5. https://doi.org/10.1109/IMPACT55510.2022.10029268

Arostegui Carrasco, A. (2022). *Mejora de la efectividad de la clasificación en la plataforma WEKA en base al uso de métodos de remuestreo sobre la distribución de clases óptima*. https://addi.ehu.es/handle/10810/58983

Awni, M., Khalil, M. I., & Abbas, H. M. (2022). Offline Arabic handwritten word recognition: A transfer learning approach. *Journal of King Saud University-Computer and Information Sciences*, *34*(10), 9654–9661. https://doi.org/10.1016/j.jksuci.2021.11.018

Ayala, M. M. S. (2017). *Aportes a la Kilka o Escritura Inka* (1ra. ed.). Juan Gutemberg Editores Impresores.

Ayyadevara, V. K., & Reddy, Y. (2020). *Modern Computer Vision with PyTorch: Explore deep learning concepts and implement over 50 real-world image applications*. Packt Publishing Ltd.

Bernico, M. (2018). *Deep Learning Quick Reference: Useful hacks for training and optimizing deep neural networks with TensorFlow and Keras*. Packt Publishing Ltd.

Boido, M. (2012). *Sincronía universal: la relación palabra-imagen en la obra de Xul*

Solar. 272. http://cdigital.uv.mx/handle/123456789/33509

Bouysse-Cassagne, T., & Harris, O. (1988). Pacha en torno al pensamiento aymara. In *Raíces de América: el mundo Aymara* (pp. 217–281). Alianza América, Unesco. https://dialnet.unirioja.es/servlet/articulo?codigo=9114679

Brownlee, J. (2019). *Deep learning for computer vision: image classification, object detection, and face recognition in python* (1.4). Machine Learning Mastery.

Burger, W., & Burge, M. J. (2010). *Principles of digital image processing: fundamental techniques*. Springer Science & Business Media.

Cabral, E. S. (2019). O RESGATE DE UMA LÍNGUA-A CONSTRUÇÃO DA ESCRITA QUÉCHUA. *Muiraquitã: Revista De Letras E Humanidades*, *7*(1). https://doi.org/10.29327/212034.7.1-4

Carreño, P. (2006). El quechua y la modernidad: instrumentos para crear un vocabulario actual. *RunasimiNet. Lima: RIDEI-Pontificia Universidad Católica Del Perú (PUCP)*. https://red.pucp.edu.pe/ridei/files/2011/08/199.pdf

Caso, A. (1989). *Alfonso Caso: de la arqueología a la antropología* (R. Pérez-Taylor (ed.); 1ra. ed., Vol. 102). UNAM.

Cerrón-Palomino, R. (1987). *Lingüística quechua* (2da.). Centro de Estudios Rurales Andinos Bartolomé de Las Casas.

Cervantes, H., & Miguel, S. (2019). *Algoritmos metaheurísticos para la segmentación de imágenes*. https://dialnet.unirioja.es/servlet/tesis?codigo=223245

Chandrakala, H. T., & Thippeswamy, G. (2020). *Deep Convolutional Neural Networks for Recognition of Historical Handwritten Kannada Characters BT - Frontiers in Intelligent Computing: Theory and Applications* (S. C. Satapathy, V. Bhateja, B. Le Nguyen, N. G. Nguyen, & D.-N. Le (eds.); pp. 69–77). Springer Singapore. https://doi.org/10.1007/978-981-13-9920-6_7

Chauhan, N., Bhatt, A. K., Dwivedi, R. K., & Belwal, R. (2018). Accuracy testing of data classification using tensor flow a python framework in ANN designing. *2018 International Conference on System Modeling & Advancement in Research Trends (SMART)*, 44–48. https://doi.org/10.1109/SYSMART.2018.8746945

Chityala, R., & Pudipeddi, S. (2020). *Image processing and acquisition using Python* (2nd ed.). CRC Press. https://doi.org/10.1201/9780429243370

Chiwariro, R., & Wosowei, J. B. (2023). Comparative analysis of deep learning convolutional neural networks based on transfer learning for pneumonia detection. *Int. J. Res. Appl. Sci. Eng. Technol*, *11*(1), 1161–1170. https://doi.org/10.22214/ijraset.2023.48685

Chung, B. W. C. (2017). *Pro Processing for Images and Computer Vision with OpenCV*. http://103.62.146.201:8081/jspui/handle/1/5448

Cobo, B. (1892). *Historia del Nuevo Mundo - Tomo III*. Imp. de E. Rasco. https://bdh-rd.bne.es/viewer.vm?id=0000044724&page=1155

Cummins, T. B. F. (2011). Tocapu: What Is It, What Does It Do, and Why Is It Not a Knot? In *Their way of writing: Scripts, signs, and pictographies in pre-Columbian America*. Dumbarton Oaks; Harvard University.

Dawson-Howe, K. (2014). *A practical introduction to computer vision with opencv*. John Wiley & Sons Ltd.

De la Jara, V. (2017). El Desciframiento De La Escritura De Los Inkas. *Arqueología y Sociedad*, *0*(7–8), 75–84. https://doi.org/10.15381/arqueolsoc.1972n7-8.e12768

Demaagd, K., Oliver, A., Oostendorp, N., & Scott, K. (2012). *Practical Computer Vision with SimpleCV: the simple way to make technology see*. O'Reilly Media, Inc.

Deore, S. P., & Pravin, A. (2020). Devanagari handwritten character recognition using fine-tuned deep convolutional neural network on trivial dataset. *Sādhanā*, *45*, 1–13. https://doi.org/10.1007/s12046-020-01484-1

Dey, S. (2018). *Hands-On Image Processing with Python: Expert techniques for advanced image analysis and effective interpretation of image data*. Packt Publishing Ltd.

di Sangro, R. (1750). *Lettera apologetica dell' esercitato Accademico della Crusca contenente la difesa del libro intitolato Lettere d'una Peruana, per rispetto alla supposizione de'quipu*. Gennaro Morelli.

Distante, A., & Distante, C. (2020). *Handbook of image processing and computer vision: Volume 2: From image to pattern*. https://doi.org/10.1007/978-3-030-42374-2

Eeckhout, P., & Danis, N. (2004). Los tocapus reales en Guamán Poma: ¿una heráldica incaica? *Boletín de Arqueología PUCP*, *8*, 305–323. https://doi.org/10.18800/boletindearqueologiapucp.200401.016

Ekman, M. (2021). *Learning deep learning: theory and practice of neural networks, computer vision, NLP, and transformers using TensorFlow*. Addison-Wesley Professional.

El-Amir, H., & Hamdy, M. (2020). Deep learning pipeline. *Building a Deep Learning Model with TensorFlow*, 114. https://doi.org/10.1007/978-1-4842-5349-6

Elleuch, M., Jraba, S., & Kherallah, M. (2021). The Effectiveness of Transfer Learning for Arabic Handwriting Recognition using Deep CNN. *Journal of Information Assurance & Security*, *16*(2). http://www.mirlabs.org/jias/secured/Volume16-Issue2/Paper8.pdf

Frame, M. (2010). What Guaman Poma shows us, but doesn't tell us, about tukapu. *Ñawpa Pacha*, *30*(1), 25–52. https://doi.org/10.1179/naw.2010.30.1.25

Frangi, A. F., Niessen, W. J., Vincken, K. L., & Viergever, M. A. (1998). Multiscale vessel enhancement filtering. *Medical Image Computing and Computer-Assisted Intervention—MICCAI'98: First International Conference Cambridge, MA, USA, October 11–13, 1998 Proceedings 1*, 130–137. https://doi.org/10.1007/bfb0056195

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*(4), 193–202. https://doi.org/10.1007/BF00344251

Furht, B., Akar, E., & Andrews, W. A. (2018). *Digital Image Processing: Practical Approach*. Springer. https://doi.org/10.1007/978-3-319-96634-2

Garcés, F., & Sánchez, W. (2014). La colección de escrituras ideográficas andinas del Instituto de Investigaciones Antropológicas y Museo de la Universidad Mayor de San Simón: una forma de narrar sin letras. *Escritura Andina: Pictografía e*

Ideografía En Cuero y Papel, 13–37. http://cendoc.chirapaq.org.pe/items/show/9311

Garcia Chilan, C., & Viteri Paredes, J. (2010). *Análisis e implementación de algoritmos para distorsionar imágenes con distintos tipos de ruido y aplicación de filtros en dos dimensiones para restaurarlas*. https://www.researchgate.net/publication/44284326

Garg, A., Tiwari, L., Juj, T., Indu, S., & Jayanthi, N. (2022). Language and Era Prediction of Digitized Indian Manuscripts Using Convolutional Neural Networks. *Sentimental Analysis and Deep Learning: Proceedings of ICSADL 2021*, 703–718. https://doi.org/10.1007/978-981-16-5157-1_55

Gentile, M. E. (2008). El Tocapu 285: Consideraciones acerca de la llamada "Escritura Incaica." *ARKEOS: Revista Electrónica de Arqueología PUCP*, *3*(2), 1–17. https://www.academia.edu/16412018/El_tocapu_285_Consideraciones_acerca_de_la_llamada_escritura_incaica_

Ghosh, T., Abedin, M.-H.-Z., Al Banna, H., Mumenin, N., & Abu Yousuf, M. (2021). Performance analysis of state of the art convolutional neural network architectures in Bangla handwritten character recognition. *Pattern Recognition and Image Analysis*, *31*, 60–71. https://doi.org/10.1134/S1054661821010089

González-Acuña, R. G., Chaparro-Romo, H. A., & Melendez-Montoya, I. (2021). *Optics and Artificial Vision*. IOP Publishing Ltd.

Gonzalez-Holguin, D. (1608). *Vocabulario de la lengua general de todo el Perú, llamada lengua quichua, o del Inca*. http://bdh-rd.bne.es/viewer.vm?id=0000049038&page=1

González, A. S. (2014). La Escritura Cifrada. *Tabularium*, *1*, 92–100. https://dialnet.unirioja.es/servlet/articulo?codigo=8193081

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. The MIT Press.

Gulli, A., & Pal, S. (2017). *Deep learning with Keras*. Packt Publishing Ltd.

Gurav, Y., Bhagat, P., Jadhav, R., & Sinha, S. (2020). Devanagari Handwritten Character Recognition using Convolutional Neural Networks. *2020 International*

Conference on Electrical, Communication, and Computer Engineering (ICECCE), 1–6. https://doi.org/10.1109/ICECCE49384.2020.9179193

Habib, G., & Qureshi, S. (2022). Optimization and acceleration of convolutional neural networks: A survey. *Journal of King Saud University-Computer and Information Sciences*, *34*(7), 4244–4268. https://doi.org/10.1016/j.jksuci.2020.10.004

Hazra, A., Choudhary, P., Inunganbi, S., & Adhikari, M. (2021). Bangla-Meitei Mayek scripts handwritten character recognition using convolutional neural network. *Applied Intelligence*, *51*(4), 2291–2311. https://doi.org/10.1007/s10489-020-01901-2

Ho, Y., & Wookey, S. (2019). The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, *8*, 4806–4813. https://doi.org/10.1109/ACCESS.2019.2962617

Hornberger, N. H., & Coronel-Molina, S. M. (2004). *Quechua language shift, maintenance, and revitalization in the Andes: The case for language planning*. https://doi.org/doi.org/10.1515/ijsl.2004.025

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708. https://openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html

Huillca-Huallpa, A. (2016). *Qhapaq Khipu Teqsi Simi*. Casa Museo de Arte y Cultura. https://es.scribd.com/document/554267761/Qhapaq-Khipu-Teqsi-Simi-a-h-h

Hyland, S., & Hyland, W. P. (2020). Secret strings: the sounds of fibre and ply. *The Material Culture of Basketry*. https://doi.org/10.5040/9781350094062.ch-012

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *ICLR 2017*, 1–13. https://doi.org/10.48550/arXiv.1602.07360

Inunganbi, S., Choudhary, P., & Manglem, K. (2021). Handwritten Meitei Mayek recognition using three-channel convolution neural network of gradients and gray.

Computational Intelligence, *37*(1), 70–86. https://doi.org/10.1111/coin.12392

Jara, V. de la. (1975). *Introducción al estudio de la escritura de los inkas*. INIDE Ediciones Previas.

Jose, B., & Pushpalatha, K. P. (2023). Malayalam Handwritten Character Recognition Using Transfer Learning. *2023 International Conference on Advances in Intelligent Computing and Applications (AICAPS)*, 1–5. https://doi.org/10.1109/AICAPS57044.2023.10074586

Karolinska, N. (1981). *The Nobel Prize in Physiology or Medicine 1981*. http://www.neurosciences.us/courses/systems/Nobels/1981 Sperry, Hubel & Wiesel/Press Release.pdf

Ketkar, N., & Moolayil, J. (2021). *Deep learning with Python: learn best practices of deep learning models with PyTorch*. Apress.

Khan, S., Nazir, S., Khan, H. U., & Hussain, A. (2021). Pashto Characters Recognition Using Multi-Class Enabled Support Vector Machine. *CMC-COMPUTERS MATERIALS & CONTINUA*, *67*(3), 2831–2844. https://doi.org/10.32604/cmc.2021.015054

Kinser, J. M. (2018). *Image Operators: Image Processing in Python*. CRC Press. https://doi.org/10.1201/9780429451188

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Krüger, F. (2016). *Activity, context, and plan recognition with computational causal behaviour models*. Universität Rostock. https://www.researchgate.net/publication/314116591

Kulkarni, A., Shivananda, A., & Sharma, N. R. (2022). *Computer Vision Projects with PyTorch*. https://doi.org/10.1007/978-1-4842-8273-1

Lang, K. (2001). The Curse of the Inca Gold. In *Discovery Channel*. https://www.youtube.com/watch?v=5YeB02LqEQA&t=1982s

Laurencich-Minelli, L. (2016a). La escritura de los Incas a la luz de dos documentos jesuíticos recién descubiertos. *Antiguos Jesuitas En Iberoamérica*, *4*(1), 68. https://doi.org/10.31057/2314.3908.v4.n1.17633

Laurencich-Minelli, L. (2016b). Quipu y escritura en el antiguo Perú. Estado de la cuestión. *FUENTES*, *10*(44), 6–24. http://revistasbolivianas.umsa.bo/scielo.php?script=sci_arttext&pid=S1997-44852016000300003

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551. https://doi.org/10.1162/neco.1989.1.4.541

Li, X. (2018). Image Interpolation. *Encyclopedia of Image Processing*, 308. https://doi.org/10.1201/9781351110273-140000467

Liu, Y. H., & Mehta, S. (2019). *Hands-On Deep Learning Architectures with Python: Create deep neural networks to solve computational problems using TensorFlow and Keras*. Packt Publishing Ltd.

Lukasik, E., Charytanowicz, M., Milosz, M., Tokovarov, M., Kaczorowska, M., Czerwinski, D., & Zientarski, T. (2021). Recognition of handwritten Latin characters with diacritics using CNN. *Bulletin of the Polish Academy of Sciences: Technical Sciences*, e136210–e136210. https://doi.org/10.24425/bpasts.2020.136210

Makitalo, M., & Foi, A. (2010). Optimal inversion of the Anscombe transformation in low-count Poisson image denoising. *IEEE Transactions on Image Processing*, *20*(1), 99–109. https://ieeexplore.ieee.org/abstract/document/5504216

Masruroh, S. U., Syahid, M. F., Munthaha, F., Muharram, A. T., & Putri, R. A. (2023). Deep Convolutional Neural Networks Transfer Learning Comparison on Arabic Handwriting Recognition System. *JOIV: International Journal on Informatics Visualization*, *7*(2), 330–337. http://www.joiv.org/index.php/joiv/article/view/1605/646

McAndrew, A. (2016). *A computational introduction to digital image processing* (Vol. 2). CRC Press Boca Raton. https://doi.org/10.1201/b19431

Moocarme, M., Abdolahnejad, M., & Bhagwat, R. (2020). *The Deep Learning with Keras Workshop: Learn how to define and train neural network models with just a few lines of code*. Packt Publishing Ltd.

Mora Lazarini, A. (2008). *Funciones Gabor bidimensionales para el análisis y clasificación de texturas*. https://1library.co/document/qo59192k-funciones-gabor-bidimensionales-análisis-clasificación-texturas.html

Mordvintsev, A., & Abid, K. (2014). *Opencv-python tutorials documentation*. https://www.kaggle.com/blobs/download/forum-message-attachment-files/16192/OpenCV-Python Tutorials-2017.pdf

Moscovich, V. (2010). *El Khipu y la Yupana*. Lima, Perú. https://www.researchgate.net/publication/319797263

Mridha, M. F., Ohi, A. Q., Ali, M. A., Emon, M. I., & Kabir, M. M. (2021). BanglaWriting: A multi-purpose offline Bangla handwriting dataset. *Data in Brief*, *34*, 106633. https://doi.org/10.1016/j.dib.2020.106633

Nanehkaran, Y. A., Zhang, D., Salimi, S., Chen, J., Tian, Y., & Al-Nabhan, N. (2021). Analysis and comparison of machine learning classifiers and deep neural networks techniques for recognition of Farsi handwritten digits. *The Journal of Supercomputing*, *77*(4), 3193–3222. https://doi.org/10.1007/s11227-020-03388-7

Neto, A. F. D. S., Bezerra, B. L. D., Lima, E. B., & Toselli, A. H. (2020). HDSR-Flor: A Robust End-to-End System to Solve the Handwritten Digit String Recognition Problem in Real Complex Scenarios. *IEEE Access*, *8*, 208543–208553. https://doi.org/10.1109/ACCESS.2020.3039003

Ochoa, L. E., & Castrillón, A. A. C. (2006). Historia y geografía: imaginarios en la literatura boliviana. *Lingüística y Literatura*, *49*, 209–214. https://www.redalyc.org/articulo.oa?id=476548927013

Parkin, A. (2018). *Computing Colour Image Processing*. Springer. https://doi.org/10.1007/978-3-319-74076-8

Pearlsy, P. V, & Sankar, D. (2023). Malayalam Handwritten Character Recognition using Transfer Learning and Fine Tuning of Deep Convolutional Neural Networks. *2023 3rd International Conference on Advances in Computing, Communication, Embedded and Secure Systems*, 125–129. https://doi.org/10.1109/ACCESS57397.2023.10200336

Peñuela, J. (2017). Camino a San Moritz: la investigación en artes es experiencia intensa de los cuerpos. *Calle 14: Revista de Investigación En El Campo Del Arte*, *12*(21), 7. https://dialnet.unirioja.es/servlet/articulo?codigo=5963892

Poma de Ayala, G. (1615). *Nueva Corónica y Buen Gobierno*. https://repositoriodigital.bnp.gob.pe/bnp/recursos/2/flippingbook/1000089657/files/assets/basic-html/page-283.html#

Pratama, S. B. (2021). *SVD Implementation on MNIST Image Classification Based on CNN*. https://doi.org/10.13140/RG.2.2.32677.22241

Purnamawati, S., Rachmawati, D., Lumanauw, G., Rahmat, R. F., & Taqyuddin, R. (2018a). Korean letter handwritten recognition using deep convolutional neural network on android platform. *Journal of Physics: Conference Series*, *978*(1). https://doi.org/10.1088/1742-6596/978/1/012112

Purnamawati, S., Rachmawati, D., Lumanauw, G., Rahmat, R. F., & Taqyuddin, R. (2018b). Korean letter handwritten recognition using deep convolutional neural network on android platform. *Journal of Physics: Conference Series*, *978*(1), 12112. https://doi.org/10.1088/1742-6596/978/1/012112

Rafael C, G., & Richard E, W. (2018). *Digital image processing* (4th ed.). Pearson Education Ltd.

Rajpal, D., & Garg, A. R. (2023a). Deep Learning Model for Recognition of Handwritten Devanagari Numerals with Low Computational Complexity and Space Requirements. *IEEE Access*. https://doi.org/10.1109/ACCESS.2023.3277392

Rajpal, D., & Garg, A. R. (2023b). Ensemble of deep learning and machine learning approach for classification of handwritten Hindi numerals. *Journal of Engineering and Applied Science*, *70*(1), 81. https://doi.org/10.1186/s44147-023-00252-2

Rajpal, D., Garg, A. R., Mahela, O. P., Alhelou, H. H., & Siano, P. (2021). A fusion-based hybrid-feature approach for recognition of unconstrained offline handwritten hindi characters. *Future Internet*, *13*(9), 239. https://doi.org/10.3390/fi13090239

Rani, N. S., Subramani, A. C., P., A. K., & Pushpa, B. R. (2020). Deep Learning Network Architecture based Kannada Handwritten Character Recognition. *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*, 213–220. https://doi.org/10.1109/ICIRCA48905.2020.9183160

Ranjan, S., & Senthamilarasu, S. (2020). *Applied Deep Learning and Computer Vision for Self-Driving Cars: Build autonomous vehicles using deep neural networks and behavior-cloning techniques*. Packt Publishing Ltd.

Rebaza, J. V. (2007). Detección de bordes mediante el algoritmo de Canny. *Escuela Académico Profesional Di Informática. Universidad Nacional de Trujillo*, *4*. https://www.researchgate.net/profile/Jorge-Valverde-Rebaza/publication/267240432

Rehman, A. (2021). *Neural Computing for Online Arabic Handwriting Recognition Using Hard Stroke Features Mining*. https://doi.org/10.48550/arXiv.2005.02171

Riaz, M., Monir, S. M. G., & Hasan, R. (2022). Evaluation of deep learning approaches for optical character recognition in Urdu language. *Mehran University Research Journal of Engineering and Technology*, *41*(4), 146–156. https://doi.org/10.22581/muet1982.2204.15

Ricardo, A. (1586). *Arte y vocabulario de la lengua general del Perú, llamada quechua, y de la lengua española* (p. 445). https://bdh.bne.es/bnesearch/detalle/4205294

Rivera, A. T. (2021). *Escrituras Andinas Quipu, Tocapu, Püron y Ñimin*. 126. https://www.academia.edu/download/65639729/ESCRITURAS_ANDINAS.pdf

Rodríguez, G. (2015). Las caligrafías heréticas de León Ferrari. *Conclusiones Analíticas*, *2*. http://sedici.unlp.edu.ar/handle/10915/48334

Roncoroni, U. (2017). Quipus, computation and generative grammars. *GA2018 – XXI Generative Art Conference*, *58*(3), 355–367. http://www.generativeart.com/GA2018_papers/GA2018_Umberto

Roncoroni.pdf

Rosebrock, A. (2016). Practical Python and OpenCV: An Introductory, Example Driven Guide to Image Processing and Computer Vision, 3rd Edition. In *Igarss 2014* (2nd ed., Issue 1). PyImageSearch.

Rosebrock, A. (2017). *Deep learning for computer vision with python: Starter bundle* (1st ed.). PyImageSearch.

Salomon, F. (2004). *The cord keepers: Khipus and cultural life in a Peruvian village*. Duke University Press. https://doi.org/10.2307/j.ctv11hpp2t

Sánchez, F. J. (2010). *Medición y Análisis de las Variaciones en el Nivel de un Modelo Físico Empleando Imágenes*. Universidad Autónoma Metropolitana, México. http://newton.azc.uam.mx/mcc/01_esp/11_tesis/tesis/Propuestas_tesis/Tesis Franciscov6.pdf

Sandhya, S., & Geetha, V. (2022). Isolated Kannada Character Recognition using Densely Connected Convolutional Network. *2022 International Conference on Asian Language Processing (IALP)*, 137–142. https://doi.org/10.1109/IALP57159.2022.9961284

Sandron, M. (1999). Un intento de lectura pictográfica e ideográfica de unos queros coloniales del Museo de América. *Anales Del Museo de América*, *7*, 141–156. https://dialnet.unirioja.es/servlet/articulo?codigo=1455912

Shady, R., Narváez, J., & López, S. (2000). La Antigüedad del Uso del Quipu como Escritura: Las Evidencias de la Huaca San Marcos. *BOLETÍN. Museo de Arqueología y Antropología*, *3*(10), 2–23. https://biblioteca.imarpe.gob.pe/opac_css/index.php?lvl=notice_display&id=26404

Shams, M., Elsonbaty, A., & ElSawy, W. (2020). Arabic Handwritten Character Recognition based on Convolution Neural Networks and Support Vector Machine. *(IJACSA) International Journal of Advanced Computer Science and Applications*, *11*(8), 144–149. https://doi.org/10.14569/IJACSA.2020.0110819

Shanmugamani, R. (2018). *Deep Learning for Computer Vision: Expert techniques to*

train advanced neural networks using TensorFlow and Keras. Packt Publishing Ltd.

Shrivakshan, G. T., & Chandrasekar, C. (2012). A comparison of various edge detection techniques used in image processing. *International Journal of Computer Science Issues (IJCSI)*, *9*(5), 269. https://www.researchgate.net/publication/303142762

Sikder, S. U., Muslebeen, M. S., Karim, D. Z., Saha, R., & Bushra, T. A. (2022). OkkhorNet: A Deep Convolutional Neural Network to Classify Bengali Handwritten Characters. *2022 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, 1–6. https://doi.org/10.1109/CSDE56538.2022.10089315

Silverman, G. (2011). La escritura Inca: la representación geométrica del Quechua Precolombino. *Ex Novo: Revista d'història i Humanitats*, 37–49. https://www.raco.cat/index.php/ExNovo/article/view/250696

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015*. https://doi.org/10.48550/arXiv.1409.1556

Singh, H. (2019). *Practical Machine Learning and Image Processing For Facial Recognition, Object Detection, and Pattern Recognition Using Python*. Springer. https://doi.org/10.1007/978-1-4842-4149-3

Solomon, C., & Breckon, T. (2011). *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons Ltd.

Sotelo, J. A. L. (2021). *Deep Learning: Teoría y aplicaciones* (1ra ed.). Alpha Editorial.

Sridevi, T. N., & Rangarajan, L. (2021). Deep convolutional neural networks for degraded printed kannada character recognition. *Indian Journal of Computer Science and Engineering*, *12*(3), 719–727. https://doi.org/10.21817/indjcse/2021/v12i3/211203187

Steckbauer, S. M. (2000). *El quechua* (pp. 57–78). Vervuert Verlagsgesellschaft. https://doi.org/doi:10.31819/9783954879762-003

Sundararajan, D. (2017). *Digital image processing: a signal processing and algorithmic*

approach. Springer. https://doi.org/10.1007/978-981-10-6113-4

Sutramiani, N. P., Suciati, N., & Siahaan, D. (2021). MAT-AGCA: Multi Augmentation Technique on small dataset for Balinese character recognition using Convolutional Neural Network. *ICT Express*, *7*(4), 521–529. https://doi.org/10.1016/j.icte.2021.04.005

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826. https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Szegedy_Rethinking_the_Inception_CVPR_2016_paper.pdf

Tanodi, B. M. (2000). *Escritura sin letras en los Andes Centrales*. https://suquia.ffyh.unc.edu.ar/handle/suquia/16678

Thakur, A. (2020). *Approaching (almost) any machine learning problem*. https://docdrop.org/download_annotation_doc/AAAMLP-569to.pdf

Thanki, R. M., & Kothari, A. M. (2019). *Digital image processing using SCILAB*. Springer. https://doi.org/10.1007/978-3-319-89533-8

Thomas, S., & Passi, S. (2019). *PyTorch Deep Learning Hands-On: Build CNNs, RNNs, GANs, Reinforcement Learning, and More, Quickly and Easily*. Packt Publishing Ltd.

Tuncer, T., Aydemir, E., Ozyurt, F., & Dogan, S. (2022). A deep feature warehouse and iterative MRMR based handwritten signature verification method. *Multimedia Tools and Applications*, 1–15. https://doi.org/10.1007/s11042-021-11726-x

Tyagi, V. (2018). *Understanding digital image processing*. CRC Press. https://doi.org/10.1201/9781315123905

Vasilev, I. (2019). *Advanced Deep Learning with Python: Design and implement advanced next-generation AI solutions using TensorFlow and PyTorch*. Packt Publishing Ltd.

Vasilev, I., Slater, D., Spacagna, G., Roelants, P., & Zocca, V. (2019). *Python Deep Learning: Exploring deep learning techniques and neural network architectures*

with Pytorch, Keras, and TensorFlow. Packt Publishing Ltd.

Vaughan, L. (2023). *Python Tools for Scientists*. No Starch Press.

Villán, A. F. (2019). *Mastering OpenCV 4 with Python: a practical guide covering topics from image processing, augmented reality to deep learning with OpenCV 4 and Python 3.7*. Packt Publishing Ltd.

Wang, Z.-R., & Du, J. (2021). Joint architecture and knowledge distillation in CNN for Chinese text recognition. *Pattern Recognition*, *111*, 107722. https://doi.org/10.1016/j.patcog.2020.107722

Wu, Y.-C., Yin, F., & Liu, C.-L. (2017). Improving handwritten Chinese text recognition using neural network language models and convolutional neural network shape models. *Pattern Recognition*, *65*, 251–264. https://doi.org/10.1016/j.patcog.2016.12.026

Xie, Y., Chen, L., & Hofmann, U. G. (2012). Reduction of periodic noise in Fourier domain optical coherence tomography images by frequency domain filtering. *Biomedical Engineering/Biomedizinische Technik*, *57*(SI-1-Track-P), 830–832. https://doi.org/10.1515/bmt-2012-4189

Yalçın, O. G., & Yalçın, O. G. (2021). Deep learning and neural networks overview. *Applied Neural Networks with Tensorflow 2: API Oriented Deep Learning with Python*, 57–80. https://doi.org/10.1007/978-1-4842-6513-0_3

Yu, F., & Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *ArXiv Preprint ArXiv:1511.07122*. https://doi.org/10.48550/arXiv.1511.07122

Zavala, J. M., Bachraty, D., & Payàs, G. (2021). El pron o quipu mapuche según fuentes coloniales y datos arqueológicos: antecedentes sobre su origen, uso y función. *Boletín Del Museo Chileno de Arte Precolombino*, *26*(1), 41–55. https://doi.org/10.4067/S0718-68942021000100041

Zhang, Q., & Skjetne, R. (2018). *Sea Ice Image Processing with MATLAB®*. CRC Press, Inc. https://doi.org/10.1201/9781351069205

Zhang, Y.-J. (2022). *A Selection of Image Processing Techniques: From Fundamentals to Research Front*. CRC Press. https://doi.org/10.1201/9781003241416

Zhong, Z., Jin, L., & Feng, Z. (2015). Multi-font printed Chinese character recognition using multi-pooling convolutional neural network. *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, 96–100. https://doi.org/10.1109/ICDAR.2015.7333733

Zhuang, Y., Liu, Q., Qiu, C., Wang, C., Ya, F., Sabbir, A., & Yan, J. (2021). A Handwritten Chinese Character Recognition based on Convolutional Neural Network and Median Filtering. *Journal of Physics: Conference Series*, *1820*(1), 12162. https://doi.org/10.1088/1742-6596/1820/1/012162

Ziótkowski, M., & Siemianowska, S. (2021). Un "cantar de gesta" inca en un quero colonial: Presentación y estudio preliminar. *Chungará (Arica)*, *53*(1), 55–80. https://doi.org/10.4067/S0717-73562021005000404

Zuidema, R. T. (2014). 'Hacer calendarios' en quipus y tejidos. Los números y su rol en el registro simultáneo del orden sociopolítico y calendárico andino en el Cuzco, Chuquibamba y Collaguas. *Sistemas de Notación Inca: Quipu y Tocapu. Actas Del Simposio Internacional*, 395–445. https://qhapaqnan.cultura.pe/articulos?page=2

ANEXOS

Anexo 1. Matriz de consistencia

PROBLEMAS	OBJETIVOS	HIPÓTESIS	VARIABLES	INDICADORES	MÉTODOS
PG: ¿Cómo desarrollar un sistema de red neuronal convolucional profunda que obtenga mejores resultados en el reconocimiento automático de la simbología base de los quipus de escritura?	**OG:** Desarrollar un sistema de red neuronal convolucional profunda que obtenga mejores resultados en el reconocimiento automático de la simbología base de los quipus de escritura.	**HG:** El desarrollo de un sistema de red neuronal convolucional profunda permite obtener mejores resultados en el reconocimiento automático de la simbología base de los quipus de escritura, superando la exactitud de otros métodos.	**Variable Independiente (VI):** Modelos *Deep Convolutional Neural Network* **Variable Dependiente (VD):** Reconocimiento automático de la simbología base de la escritura Inca.	- Función de pérdida - Exactitud - Tiempo de entrenamiento - Matriz de confusión - Peso o tamaño del modelo	**Tipo de investigación:** - Aplicada, Explicativa **Método:** - Cuantitativo **Diseño de investigación:** - Longitudinal **Paradigma:** - Positivista **Población:** - 167 clases **Muestra:** - 40 clases **Pruebas Estadísticas:** - Análisis descriptivo - Aprendizaje automático
PE1: ¿Cómo es el sistema de escritura Inca, particularmente los quipus de escritura y determinar sus posibles patrones?	**OE1:** Descifrar el sistema de escritura Inca, particularmente los quipus de escritura y sus posibles patrones.	**HE1:** El sistema de escritura Inca se basa en los quipus de escritura que es simbólica, pictográfica, ideográfica, geométrica, abstracta, logosilábica y cifrada; tiene patrones muy definidos.		- Número de símbolos utilizados en los quipus de escritura - Formas de los símbolos base - Número de patrones identificados - Número de palabras y sílabas - Promedio de nudos por cuerda	
PE2: ¿Cómo construir un conjunto de datos significativo de imágenes de los quipus de escritura que permita entrenar modelos mediante aprendizaje por transferencia?	**OE2:** Construir un conjunto de datos significativo de imágenes de los quipus de escritura para entrenar modelos mediante aprendizaje por transferencia.	**HE2:** La construcción de un conjunto de datos significativo de imágenes de los quipus de escritura permite entrenar modelos mediante aprendizaje por transferencia de manera efectiva.		- Probabilidad asociada a la clase (imagen digital) - Probabilidad asociada a la clase (en tiempo real)	
PE3: ¿Cuál es el mejor modelo de redes neuronales convolucionales entrenado evaluado bajo las mismas condiciones y métricas en la tarea del reconocimiento de la simbología base de los quipus de escritura?	**OE3:** Evaluar los modelos de redes neuronales convolucionales entrenados bajos las mismas condiciones y métricas; y determinar el mejor modelo en la tarea de reconocimiento de la simbología base de los quipus de escritura.	**HE3:** El modelo de red neuronal convolucional profunda *DenseNet121* mejora significativamente la exactitud en el reconocimiento automático de la simbología base de los quipus de escritura en comparación con otros métodos.		- Funciones de pérdida - Exactitudes - Tiempos de entrenamiento - Pesos o tamaños de los modelos	

Anexo 2. Código fuente para el entrenamiento

Training

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

from __future__ import division
from __future__ import print_function

import copy
import os
import time
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import matplotlib.pyplot as plt
import numpy as np
from torchvision import datasets, models, transforms
from datetime import datetime
from pathlib import Path

print("PyTorch Version: ", torch.__version__)
print("Torchvision Version: ", torchvision.__version__)
```

```
PyTorch Version:  1.9.0+cu102
Torchvision Version:  0.10.0+cu102
```

```
def train_model(model, dataloaders, criterion, optimizer, save_path,
num_epochs=25, is_inception=False):
    Path(save_path).mkdir(parents=True, exist_ok=True)
    since = time.time()

    acc_train_history = []
    loss_train_history = []
    acc_val_history = []
    loss_val_history = []

    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    for epoch in range(num_epochs):
        print('Epoch {}/{}'.format(epoch, num_epochs - 1))
        print('-' * 10)

        for phase in ['train', 'val']:
            if phase == 'train':
                model.train()
            else:
                model.eval()

            running_loss = 0.0
            running_corrects = 0

            for inputs, labels in dataloaders[phase]:
                inputs = inputs.to(device)
                labels = labels.to(device)

                optimizer.zero_grad()
```

```
                with torch.set_grad_enabled(phase == 'train'):
                    if is_inception and phase == 'train':
                        outputs, aux_outputs = model(inputs)
                        loss1 = criterion(outputs, labels)
                        loss2 = criterion(aux_outputs, labels)
                        loss = loss1 + 0.4 * loss2
                    else:
                        outputs = model(inputs)
                        loss = criterion(outputs, labels)

                    _, preds = torch.max(outputs, 1)

                    if phase == 'train':
                        loss.backward()
                        optimizer.step()

                running_loss += loss.item() * inputs.size(0)
                running_corrects += torch.sum(preds == labels.data)

            epoch_loss = running_loss /
len(dataloaders[phase].dataset)
            epoch_acc = running_corrects.double() /
len(dataloaders[phase].dataset)

            print('{} Loss: {:.4f} Acc: {:.4f}'.format(phase,
epoch_loss, epoch_acc))

            if phase == 'val' and epoch_acc > best_acc:
                best_acc = epoch_acc
                best_model_wts = copy.deepcopy(model.state_dict())
                torch.save({'epoch': epoch, 'model_state_dict':
model.state_dict(),
                            'optimizer_state_dict':
optimizer.state_dict(), 'loss': loss},
                           f'{save_path}/best.pt')
            if phase == 'val':
                loss_val_history.append(epoch_loss)
                acc_val_history.append(epoch_acc)

            if phase == 'train':
                loss_train_history.append(epoch_loss)
                acc_train_history.append(epoch_acc)

            if epoch % 10 == 0:
                torch.save({'epoch': epoch, 'model_state_dict':
model.state_dict(),
                            'optimizer_state_dict':
optimizer.state_dict(), 'loss': loss},
                           f'{save_path}/model_{epoch}.pt')
            torch.save(
                {'epoch': epoch, 'model_state_dict':
model.state_dict(), 'optimizer_state_dict': optimizer.state_dict(),
                 'loss': loss},
                f'{save_path}/last.pt')

    time_elapsed = time.time() - since
    print('Training complete in {:.0f}m {:.0f}s'.format(time_elapsed
// 60, time_elapsed % 60))
    print('Best val Acc: {:4f}'.format(best_acc))
```

```
    N = num_epochs
    plt.style.use("ggplot")
    fig = plt.figure()
    plt.plot(np.arange(0, N), acc_train_history, label="train_acc")
    plt.plot(np.arange(0, N), acc_val_history, label="val_acc")
    plt.title("Training Accuracy")
    plt.xlabel("Epochs")
    plt.ylabel("Accuracy")
    plt.legend(loc="lower right")
    fig.savefig(f'Accuracy_{model_name}.png')

    plt.style.use("ggplot")
    fig = plt.figure()
    plt.plot(np.arange(0, N), loss_train_history, label="train_loss")
    plt.plot(np.arange(0, N), loss_val_history, label="val_loss")
    plt.title("Training Loss")
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.legend(loc="upper right")
    fig.savefig(f'Loss_{model_name}.png')

    model.load_state_dict(best_model_wts)
    return model, acc_val_history

def set_parameter_requires_grad(model, feature_extracting):
    if feature_extracting:
        for param in model.parameters():
            param.requires_grad = False

def initialize_model(model_name, num_classes, feature_extract,
use_pretrained=True):
    model_ft = None
    input_size = 0

    if model_name == "AlexNet":
        """ Alexnet
        """
        model_ft = models.alexnet(pretrained=use_pretrained)
        set_parameter_requires_grad(model_ft, feature_extract)
        num_ftrs = model_ft.classifier[6].in_features
        model_ft.classifier[6] = nn.Linear(num_ftrs, num_classes)
        input_size = 224

    elif model_name == "VGG11-BN":
        """ VGG11_bn
        """
        model_ft = models.vgg11_bn(pretrained=use_pretrained)
        set_parameter_requires_grad(model_ft, feature_extract)
        num_ftrs = model_ft.classifier[6].in_features
        model_ft.classifier[6] = nn.Linear(num_ftrs, num_classes)
        input_size = 224

    elif model_name == "SqueezeNet":
        """ Squeezenet
        """
        model_ft = models.squeezenet1_0(pretrained=use_pretrained)
        set_parameter_requires_grad(model_ft, feature_extract)
        model_ft.classifier[1] = nn.Conv2d(512, num_classes,
kernel_size=(1, 1), stride=(1, 1))
        model_ft.num_classes = num_classes
        input_size = 224
```

```
    elif model_name == "DenseNet":
        """ Densenet
        """
        model_ft = models.densenet121(pretrained=use_pretrained)
        set_parameter_requires_grad(model_ft, feature_extract)
        num_ftrs = model_ft.classifier.in_features
        model_ft.classifier = nn.Linear(num_ftrs, num_classes)
        input_size = 224

    elif model_name == "Inceptionv3":
        """ Inception v3
        """
        model_ft = models.inception_v3(pretrained=use_pretrained)
        set_parameter_requires_grad(model_ft, feature_extract)
        num_ftrs = model_ft.AuxLogits.fc.in_features
        model_ft.AuxLogits.fc = nn.Linear(num_ftrs, num_classes)
        num_ftrs = model_ft.fc.in_features
        model_ft.fc = nn.Linear(num_ftrs, num_classes)
        input_size = 299

    else:
        print("Invalid model name, exiting...")
        exit()

    return model_ft, input_size

data_dir = "/content/drive/MyDrive/dataset"

# Modelos [AlexNet, VGG11-BN, SqueezeNet, DenseNet, Inceptionv3]
model_name = "DenseNet"

num_classes = 40

batch_size = 8

num_epochs = 100

mode = 'test'

feature_extract = False

model_ft, input_size = initialize_model(model_name, num_classes,
feature_extract, use_pretrained=True)

print(model_ft)

data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((input_size, input_size)),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224,
0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize((input_size, input_size)),
        transforms.CenterCrop(input_size),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224,
0.225])
    ]),
```

```
}
device = torch.device("cuda:0" if torch.cuda.is_available() else
"cpu")

model_ft = model_ft.to(device)

print("Initializing Datasets and Dataloaders...")

image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x),
data_transforms[x]) for x in ['train', 'val']}

dataloaders_dict = {
    x: torch.utils.data.DataLoader(image_datasets[x],
batch_size=batch_size, shuffle=True, num_workers=0) for x in
    ['train', 'val']}

params_to_update = model_ft.parameters()
print("Params to learn:")
if feature_extract:
    params_to_update = []
    for name, param in model_ft.named_parameters():
        if param.requires_grad == True:
            params_to_update.append(param)
            print("\t", name)
else:
    for name, param in model_ft.named_parameters():
        if param.requires_grad == True:
            print("\t", name)

optimizer_ft = optim.SGD(params_to_update, lr=0.001, momentum=0.9)
criterion = nn.CrossEntropyLoss()

save_path = datetime.now().strftime("%d-%m-%Y_%H%M")

model_ft, hist = train_model(model_ft, dataloaders_dict, criterion,
optimizer_ft, save_path=save_path,num_epochs=num_epochs,
is_inception=(model_name == "Inceptionv3"))
```

Anexo 3. Código fuente para la matriz de confusión

Confusion Matrix

```
import torch
from torchvision import models, datasets, transforms
from torch.utils.data import DataLoader
from sklearn.metrics import confusion_matrix
import numpy as np
model_ft.eval()
model_ft.to(device)
model_ft.load_state_dict(torch.load('/content/drive/MyDrive/Modelo/bes
t.pt')['model_state_dict'])

transform = transforms.Compose([transforms.Resize(256),
                                transforms.CenterCrop(224),
                                transforms.ToTensor(),
                                transforms.Normalize(mean=[0.485,
0.456, 0.406], std=[0.229, 0.224, 0.225])])

data_dir = '/content/drive/MyDrive/dataset/val'
test_dataset = datasets.ImageFolder(os.path.join(data_dir),
transform=transform)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)


device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

model_ft.to(device)
model_ft.eval()
y_true = []
y_pred = []

with torch.no_grad():
    for inputs, labels in test_loader:
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = model_ft(inputs)
        _, predicted = torch.max(outputs, 1)
        y_true.extend(labels.cpu().numpy())
        y_pred.extend(predicted.cpu().numpy())

conf_matrix = confusion_matrix(y_true, y_pred)

print("Matriz de Confusión:")
print(conf_matrix)
```

```
import seaborn as sn
import pandas as pd
```

```
import matplotlib.pyplot as plt

df_cm = pd.DataFrame(conf_matrix, range(40), range(40))
fig = plt.figure(figsize=(40,40))
sn.set(font_scale=1.4)

sn.heatmap(df_cm, annot=True, linewidths=0.5, linecolor="w",
annot_kws={"size": 16},xticklabels=['amaru', 'auqui', 'catollay',
'chaska', 'cituk', 'coya', 'cuntur', 'curaca', 'hananpacha', 'hipuy',
'illapa', 'inca', 'inti', 'killa', 'kuychi', 'llautu', 'mamacora',
'manquqhapaq', 'maytiñu', 'muncaynim', 'ñusta', 'oqllo', 'pachakamaq',
'pinunsun', 'puma', 'qantutika', 'quinquir', 'quyllur', 'runa',
'sinchiroca', 'suri', 'tacvehirac', 'tuta', 'ucumari', 'ukhupacha',
'unuy', 'uritu', 'uturuncu', 'wiraqocha',
'yanrinuy'],yticklabels=['amaru', 'auqui', 'catollay', 'chaska',
'cituk', 'coya', 'cuntur', 'curaca', 'hananpacha', 'hipuy', 'illapa',
'inca', 'inti', 'killa', 'kuychi', 'llautu', 'mamacora',
'manquqhapaq', 'maytiñu', 'muncaynim', 'ñusta', 'oqllo', 'pachakamaq',
'pinunsun', 'puma', 'qantutika', 'quinquir', 'quyllur', 'runa',
'sinchiroca', 'suri', 'tacvehirac', 'tuta', 'ucumari', 'ukhupacha',
'unuy', 'uritu', 'uturuncu', 'wiraqocha', 'yanrinuy'], cmap='Blues')
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
fig.savefig(f'Confusion matrix.png')
plt.show()
```

Anexo 4. Código fuente para la predicción

Prediction

```
import cv2
from copy import deepcopy
from PIL import Image
from skimage import io, transform
from google.colab.patches import cv2_imshow

model_ft.eval()
model_ft.to(device)
model_ft.load_state_dict(torch.load('/content/drive/MyDrive/dataset/be
st.pt')['model_state_dict'])

<All keys matched successfully>

data_transform = transforms.Compose([
  transforms.Resize(input_size),
  transforms.CenterCrop(input_size),
  transforms.ToTensor(),
  transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

path_img = '/content/drive/MyDrive/dataset/prueba.jpg'
if __name__ == "__main__":
  img = cv2.imread(path_img)
  cv2_imshow(img)
  img = cv2.cvtColor(deepcopy(img), cv2.COLOR_BGR2RGB)
  image_pil = Image.fromarray(img)
  img = data_transform(image_pil)
  img = img.unsqueeze(0).to(device)
  out = model_ft(img).cpu().detach().tolist()[0]
  amaru, auqui, catollay, chaska, cituk, coya, cuntur, curaca,
hananpacha, hipuy, illapa, inca, inti, killa, kuychi, llautu,
mamacora, manquqhapaq, maytiñu, muncaynim, ñusta, oqllo, pachakamaq,
pinunsun, puma, qantutika, quinquir, quyllur, runa, sinchiroca, suri,
tacvehirac, tuta, ucumari, ukhupacha, unuy, uritu, uturuncu,
wiraqocha, yanrinuy =
torch.nn.functional.softmax(torch.tensor(out)).tolist()
  symbols = {amaru:"amaru", auqui:"auqui", catollay:"catollay",
chaska:"chaska", cituk:"cituk", coya:"coya", cuntur:"cuntur",
curaca:"curaca", hananpacha:"hananpacha", hipuy:"hipuy",
illapa:"illapa", inca:"inca", inti:"inti", killa:"killa",
kuychi:"kuychi", llautu:"llautu", mamacora:"mama cora",
manquqhapaq:"manco capac", maytiñu:"maytiñu", muncaynim:"muncaynim",
ñusta:"ñusta", oqllo:"oqllo", pachakamaq:"pachakamaq",
pinunsun:"pinunsun", puma:"puma", qantutika:"qantutika",
quinquir:"quinquir", quyllur:"quyllur", runa:"runa",
sinchiroca:"sinchi roca", suri:"suri", tacvehirac:"tacvehirac",
tuta:"tuta", ucumari:"ucumari", ukhupacha:"ukhupacha", unuy:"unuy",
uritu:"uritu", uturuncu:"uturuncu", wiraqocha:"wiraqocha",
yanrinuy:"yanrinuy"}
  maxim = max(torch.nn.functional.softmax(torch.tensor(out)).tolist())
  print(symbols.get(maxim), maxim * 100, "%")
  cv2.waitKey(0)
```

Anexo 5. Código fuente para el reconocimiento en tiempo real

Real time recognition

```
import cv2
import torch
import torchvision.transforms as T
from torchvision.models import densenet121
from torchvision import utils
import numpy as np

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

model_ft.eval()
model_ft.to(device)
model_ft.load_state_dict(torch.load('best.pt')['model_state_dict'])

preprocess = T.Compose([
    T.ToPILImage(),
    T.Resize(256),
    T.CenterCrop(224),
    T.ToTensor(),
    T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
0.225]),
])

with open("C:/Users/Lenin/imagenet_classes.txt") as f:
    classes = [line.strip() for line in f.readlines()]

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()

    if not ret:
        break

    img = preprocess(frame).unsqueeze(0).to(device)

    with torch.no_grad():
        output = model_ft(img)

    _, predicted_idx = torch.max(output, 1)
    predicted_label = classes[predicted_idx.item()]

    out = model_ft(img).cpu().detach().tolist()[0]
    maxim =
max(torch.nn.functional.softmax(torch.tensor(out)).tolist())
    porcentaje = '{0:.2f}'.format(maxim * 100)+'%'
```

```
    cv2.putText(frame, f'Simbolo: {predicted_label} {porcentaje}',
(10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    cv2.imshow('Real time recognition', frame)

    if cv2.waitKey(1) & 0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Anexo 6. Matriz de confusión

Enlace de descarga

Anexo 7. Simbología base de los quipus de escritura

Enlace de descarga

Anexo 8. Sílabas generadas con repetición y sin repetición

Enlace de descarga

DECLARACIÓN JURADA DE AUTENTICIDAD DE TESIS

Por el presente documento, Yo **LENIN HUAYTA FLORES** identificado(a) con N° DNI: **80026530** en mi condición de egresado(a) del:
DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN
con código de matrícula N° 186324, informo que he elaborado la tesis denominada:
"RECONOCIMIENTO AUTOMÁTICO DE LA SIMBOLOGÍA BASE DE LA ESCRITURA INCA UTILIZANDO MODELOS DEEP CONVOLUTIONAL NEURAL NETWORK".

Es un tema original.

Declaro que el presente trabajo de tesis es elaborado por mi persona y no existe plagio/copia de ninguna naturaleza, en especial de otro documento de investigación (tesis, revista, texto, congreso, o similar) presentado por persona natural o jurídica alguna ante instituciones académicas, profesionales, de investigación o similares, en el país o en el extranjero.

Dejo constancia que las citas de otros autores han sido debidamente identificadas en el trabajo de investigación, por lo que no asumiré como suyas las opiniones vertidas por terceros, ya sea de fuentes encontradas en medios escritos, digitales o Internet.

Asimismo, ratifico que soy plenamente consciente de todo el contenido de la tesis y asumo la responsabilidad de cualquier error u omisión en el documento, así como de las connotaciones éticas y legales involucradas.

En caso de incumplimiento de esta declaración, me someto a las disposiciones legales vigentes y a las sanciones correspondientes de igual forma me someto a las sanciones establecidas en las Directivas y otras normas internas, así como las que me alcancen del Código Civil y Normas Legales conexas por el incumplimiento del presente compromiso

Puno, 04 de Septiembre del 2024.

FIRMA (Obligatorio)

Huella

Universidad Nacional del Altiplano Puno | Vicerrectorado de Investigación | Repositorio Institucional

AUTORIZACIÓN PARA EL DEPÓSITO DE TESIS O TRABAJO DE INVESTIGACIÓN EN EL REPOSITORIO INSTITUCIONAL

Por el presente documento, Yo **LENIN HUAYTA FLORES** identificado(a) con N° DNI: **80026530**, en mi condición de egresado(a) del **Programa de Maestría o Doctorado:**
DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN,
informo que he elaborado la tesis denominada:

"RECONOCIMIENTO AUTOMÁTICO DE LA SIMBOLOGÍA BASE DE LA ESCRITURA INCA UTILIZANDO MODELOS DEEP CONVOLUTIONAL NEURAL NETWORK".

para la obtención de ☒ **Grado.**

Por medio del presente documento, afirmo y garantizo ser el legítimo, único y exclusivo titular de todos los derechos de propiedad intelectual sobre los documentos arriba mencionados, las obras, los contenidos, los productos y/o las creaciones en general (en adelante, los "Contenidos") que serán incluidos en el repositorio institucional de la Universidad Nacional del Altiplano de Puno.

También, doy seguridad de que los contenidos entregados se encuentran libres de toda contraseña, restricción o medida tecnológica de protección, con la finalidad de permitir que se puedan leer, descargar, reproducir, distribuir, imprimir, buscar y enlazar los textos completos, sin limitación alguna.

Autorizo a la Universidad Nacional del Altiplano de Puno a publicar los Contenidos en el Repositorio Institucional y, en consecuencia, en el Repositorio Nacional Digital de Ciencia, Tecnología e Innovación de Acceso Abierto, sobre la base de lo establecido en la Ley N° 30035, sus normas reglamentarias, modificatorias, sustitutorias y conexas, y de acuerdo con las políticas de acceso abierto que la Universidad aplique en relación con sus Repositorios Institucionales. Autorizo expresamente toda consulta y uso de los Contenidos, por parte de cualquier persona, por el tiempo de duración de los derechos patrimoniales de autor y derechos conexos, a título gratuito y a nivel mundial.

En consecuencia, la Universidad tendrá la posibilidad de divulgar y difundir los Contenidos, de manera total o parcial, sin limitación alguna y sin derecho a pago de contraprestación, remuneración ni regalía alguna a favor mío; en los medios, canales y plataformas que la Universidad y/o el Estado de la República del Perú determinen, a nivel mundial, sin restricción geográfica alguna y de manera indefinida, pudiendo crear y/o extraer los metadatos sobre los Contenidos, e incluir los Contenidos en los índices y buscadores que estimen necesarios para promover su difusión.

Autorizo que los Contenidos sean puestos a disposición del público a través de la siguiente licencia:

Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional. Para ver una copia de esta licencia, visita: https://creativecommons.org/licenses/by-nc-sa/4.0/

En señal de conformidad, suscribo el presente documento.

Puno, 04 de Septiembre del 2024.

FIRMA (Obligatorio)

Huella