UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

EVALUACIÓN DE VULNERABILIDAD DEL CAPTCHA MEDIANTE EL RECONOCIMIENTO CON REDES NEURONALES CONVOLUCIONALES

TESIS

PRESENTADA POR:

Bach. JHON CARLOS GONZALES QUILCA

Bach. ELMER ANDER CHALCO HUARACHI

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO DE SISTEMAS

PUNO – PERÚ

2024

Reporte de similitud

NOMBRE DEL TRABAJO

EVALUACIÓN DE VULNERABILIDAD DEL CAPTCHA MEDIANTE EL RECONOCIMIENTO CON REDES NEURONALES CONVOLUCIONALES

AUTOR

JHON CARLOS GONZALES QUILCA - EL MER ANDER CHALCO HUARACHI -

RECUENTO DE PALABRAS

14592 Words

RECUENTO DE CARACTERES

82486 Characters

RECUENTO DE PÁGINAS

97 Pages

TAMAÑO DEL ARCHIVO

6.7MB

FECHA DE ENTREGA

Oct 25, 2024 10:27 AM GMT-5

FECHA DEL INFORME

Oct 25, 2024 10:29 AM GMT-5

10% de similitud general

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

- 8% Base de datos de Internet
- 2% 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 12 palabras)

V°B°

Firmado digitalmente por SOTOMAYOR ALZAMORA Guina Guadalupe FAU 20145496170 hard
Motivo: Doy V°B°
Fecha: 25.10.2024 16:30:58 -05:00

FIRMA DIGITAL

Firmado digitalmente por: VILCA HUAYTA OLIVER AMADEO
Motivo: Soy el autor del documento
Fecha: 25/10/2024 16:28:03-0500

Resumen

DEDICATORIA

Esta tesis se la dedico a mi padre celestial, por ser el redentor de todas las cosas

Esta tesis es el fruto de muchos años de esfuerzo, dedicación y sacrificio. Dedico a mis padres por su comprensión, apoyo, consejos y ayuda en los buenos y malos momentos.

A mi familia, cuyo amor incondicional y apoyo constante han sido el pilar de mi existencia. Esta obra está dedicada a ustedes, con todo mi amor y gratitud.

En memoria de mi querida hija Beyra Naiby, hoy y siempre aunque la vida nos haya puesto en situaciones difíciles, tu fortaleza y tu espíritu ha sido mi inspiración constante, llevare en nuestra memoria la luz que encendiste en mi vida, una alegría infinita, un alma llena de bondad y un amor que no se apaga con el tiempo, quiero que sepas que sin importar las circunstancias, la vida tiene una belleza infinita que siempre se revela a quienes nunca dejan de luchar, eres y serás mi mayor motivación y razón para seguir adelante. incluso en los momentos más oscuros iluminaras mi camino.

Elmer Ander Chalco Huarachi

DEDICATORIA

Dedico a Dios por derramar bendiciones sobre mí, que me escucha mediante mis oraciones y que en cada caída siempre me acompañará y llenarme de fuerzas para enfrentar cada obstáculo que se presente.

Dedico a mis padres Marcos y Julia que, con su esfuerzo y dedicación, que me dan su apoyo incondicional, por estar siempre a mi lado, por brindarme mucho afecto, cariño, paciencia, darme el motivo para seguir adelante y que siempre tendrán mi agradecimiento eterno por todo ello.

A mi hermano Alexzander, Cristian Cc. Quilca, mis tías y tíos, mis primos que son personas excepcionales que compartimos momentos únicos e inigualables, gracias por tanto cariño que me dan.

A la memoria de mi prima Geovanna Cc. Quilca, mis tíos Mario Mendez, Julián Quilca. mi tía Virginia Gonzales y demás familiares, que pasaron por mi vida en las cuales les rindo homenaje y que me tenían el aprecio y cariño que ahora se encuentran de la mano del Señor Todopoderoso.

A Criss Choque, Lenin Condori, a mi maestro Víctor Sumi, a mi maestra y madrina Angela Pinazo, mi padrino Florencio Cartagena, a Almendra E. C. Mendez y a mis amigos, que me acompañaron y tener su apoyo incondicional, gracias, por tanto, que siempre llevaré con mucha estima y consideración.

"Fuimos perfectamente imperfectos, pero a la vez fuimos ambos negativos. Las leyes de la física dicen que dos polos iguales se repelen, pero las reglas de las matemáticas dicen que negativo por negativo igualaba a positivo." Cita a Flor M. Salvador.

Jhon Carlos Gonzales Quilca

AGRADECIMIENTOS

En primer lugar, agradecer a Dios, por ser el ser supremo que siempre nos guía para alcanzar este sueño añorado.

Agradecer a la Universidad Nacional del Altiplano, nuestra Alma Mater, por brindar vuestros claustros por estos años de estudios de pregrado y posgrado, para contribuir profesionalmente en la sociedad con ética y valores.

Agradecer a la Escuela Profesional de Ingeniería de Sistemas, a las autoridades, a los docentes y personal administrativo que conforman la familia sistémica, sus instalaciones donde adquirimos conocimientos, experiencias y recuerdos que no se olvidaran de que fuimos orgullosos y felices por pertenecer y cumplir nuestra meta de ser ingenieros de sistemas.

Agradecer de manera especial a nuestro director de tesis D. Sc. Oliver Amadeo Vilca Huayta por sus consejos y sus orientaciones en este proceso de investigación, sus motivaciones y ser la persona que es.

Agradecer de manera cordial a nuestros miembros jurados de esta presente tesis, Dra. Milder Zanabria, D. Sc. Edwin Calderon y Dra. Zulema Mamani por sus consejos y sus recomendaciones en este proceso de investigación.

Agradecer a los compañeros y amigos que pasamos durante cinco años de estudios universitarios, con que pasamos años de amistad, experiencias vividas que serán únicas en el camino de la vida que siempre será un recuerdo inolvidable para cada uno de nosotros.

Jhon Carlos Gonzales Quilca

Elmer Ander Chalco Huarachi

ÍNDICE GENERAL

Pág.

Área: Inteligencia Artificial

Tema: CNN & Deep Learning

FECHA DE SUSTENTACIÓN: 29 de octubre del 2024

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE ANEXOS

Pág.

ACRÓNIMOS

IA:	Inteligencia Artificial
IC:	Ingeniería de Conocimiento
CAPTCHA:	Completely Automated Public Turing test to tell Computers and Humans Apart
PCD:	Persona con Discapacidad
CNN:	Convolutional Neural Networks
API:	Aplication Programming Interface
OpenCV:	Open Computer Vision
IDE	Entorno de Desarrollo Integrado
MVC:	Model View Control
CTV	Connectionist Temporal Classification

RESUMEN

La Prueba de Turing pública completamente automatizada para distinguir computadoras de humanos CAPTCHA (en inglés*: Completely Automated Public Turing Test to Tell Computers and Human Apart*) es una prueba que garantiza que los usuarios son personas y no computadoras. Han sido diseñadas para que las personas las puedan resolver fácilmente, pero que son difíciles para las computadoras, se utiliza para evitar que los programas informáticos con procedimientos reiterativos (bots) intenten acceder de manera no autorizada. Aunque existen varios tipos de captchas basados en texto, audio, imágenes, videos y acertijos. Muchos sitios web, incluido los servicios gubernamentales, todavía utilizan captchas basados en texto. El propósito principal de esta investigación fue desarrollar un modelo para reconocer captchas de texto de la librería GregwarCaptcha comparando dos modelos, el primero con reconocimiento óptico de imágenes y el segundo con redes neuronales convolucionales. Se obtuvo una base de datos de 1017 imágenes de captchas; seguidamente, se realizaron el preprocesamiento de datos, clasificación y entrenamiento de los dos modelos. Asimismo, se evaluó y se comparó la utilidad de los algoritmos mediante la matriz de confusión y el método F1-score, desarrollado en Python. El enfoque de investigación es cuantitativo, de nivel proyectiva. Los resultados mostraron que las redes neuronales convolucionales obtuvieron una exactitud del 97% superando significativamente al 37% del método OCR. En conclusión, esta investigación demostró la eficacia de las redes neuronales convolucionales para reconocer captchas de texto, que pueden mejorar la seguridad de los sistemas de información en un futuro.

Palabras Clave: Aprendizaje profundo, Captcha, Reconocimiento de imágenes, Redes neuronales convolucionales, Seguridad informática.

ABSTRACT

The *Completely Automated Public Turing Test to Tell Computers and Human Apart* is a test that guarantees that users are people and not computers. They have been designed so that people can solve them easily, but they are difficult for computers, it is used to prevent computer programs with repetitive procedures (bots) from trying to access them in an unauthorized way. Although there are several types of captchas based on text, audio, images, videos, and puzzles. Many websites, including government services, still use text-based captchas. The main purpose of this research was to develop a model to recognize text captchas from the GregwarCaptcha library by comparing two models, the first with optical image recognition and the second with convolutional neural networks. A database of 1017 captcha images was obtained; Next, data preprocessing, classification and training of the two models were carried out. Likewise, the usefulness of the algorithms was evaluated and compared using the confusion matrix and the F1-score method, developed in Python. The research approach is quantitative, projective. The results showed that the convolutional neural networks obtained an accuracy of 97%, significantly surpassing the 37% of the OCR method. In conclusion, this research demonstrated the effectiveness of convolutional neural networks to recognize text captchas, which can improve the security of information systems in the future.

Keywords: Captcha, Computer Security, Convolutional Neural Network, Deep learning, Image recognition.

CAPÍTULO I

INTRODUCCIÓN

1.1. PLANTEAMIENTO DEL PROBLEMA

En Internet hay una gran cantidad de sitios y servicios online, por este motivo se ha establecido diversos métodos y técnicas para proteger la información de los ataques, en especial los programas informáticos automatizados (bots) que actúan como humanos.

En primer lugar, los sistemas de CAPTCHAS también se utilizan para proteger los derechos de privacidad y seguridad de los usuarios. Si los sistemas de CAPTCHA son vulnerables, los atacantes pueden obtener acceso no autorizado a la información personal de los usuarios, lo que puede tener graves consecuencias para su privacidad y seguridad.

Por otro lado, los sistemas de CAPTCHA garantizan la accesibilidad para personas con discapacidad (PCD) especialmente para aquellos con discapacidades visuales. Si los sistemas de CAPTCHA son vulnerables, las personas con discapacidad pueden tener dificultades para acceder a los servicios en línea protegidos por estos sistemas.

Del mismo modo, las instituciones del estado peruano también hacen el uso de los captchas con un fin de proteger sus propios sistemas de información. En caso específico vamos a abordar la problemática del uso de la librería Gregwar Captcha.

Es por eso que surge la necesidad de evaluar los captchas diseñado a partir de la librería, con esta investigación vamos a desarrollar modelos mediante reconocimiento óptico por imágenes y con redes neuronales convolucionales que permitan dar los resultados de la implementación de dicho modelo y su respectiva evaluación de los captchas que provienen de la librería Gregwar Captcha.

1.2. FORMULACIÓN DE PROBLEMA

1.2.1. Problema general

¿El modelo de reconocimiento con redes neuronales convolucionales permitirá la evaluación de vulnerabilidad de un captcha?

1.2.2. Problemas específicos

- ¿Es posible construir una base de datos de entrenamiento de un modelo de reconocimiento de captcha mediante redes neuronales convolucionales?
- ¿Es posible implementar un modelo basado en el reconocimiento de captcha con redes neuronales convolucionales?
- ¿La precisión del modelo de reconocimiento de captcha con redes neuronales convolucionales es mayor que 0.9?

1.3. JUSTIFICACIÓN DE LA INVESTIGACIÓN

Con el crecimiento de Internet y sus aplicaciones en sistemas de información en comercio electrónico, defensa, educación, salud, etc. La necesidad de éstos de ser accedidos a través de internet para brindar información oportuna para la toma de decisiones en tiempo real; dichos sistemas están expuestos a ataques que se han vueltos más comunes y sofisticados, surge la necesidad de protegerlo con captchas que evitan intentos reiterados por partes de los programas sistemáticos reiterativos (bots) y personas.

Es así que los modelos tradicionales consultados para resolver los captchas contienen los siguientes pasos:

- Pre procesamiento
- Segmentación

- Post segmentación
- Reconocimiento
- Post procesamiento

Para evaluar la efectividad real de los ataques propuestos, el primer paso es medir la precisión, es decir, la fracción de captcha que pueden ser resueltas adecuadamente, pero dado su constitución y diseño se presenta diversos resultados a los análisis y adicionalmente su evaluación de vulnerabilidad, que depende de las características propias y de la confianza del atacante para resolver determinado captcha sin cambiarlo por otro lado más fácil.

Este trabajo de investigación propone una alternativa de solución mediante el uso de modelo de redes neuronales convolucionales como un método para procesar lo anteriormente argumentado en la evaluación de vulnerabilidades que se presentan en un captcha, aplicando un modelo en base a un algoritmo de aprendizaje automático profundo, una red neuronal convolucional (CNN) está especialmente diseñada para el procesamiento de dos dimensiones sobre todo en el reconocimiento de imágenes en espectrogramas también pueden ser modificados para la realización de tareas en una o varias dimensiones.

1.4. OBJETIVO DE LA INVESTIGACIÓN

1.4.1. Objetivo general

Desarrollar un modelo para reconocimiento de captcha mediante redes neuronales convolucionales que permita su evaluación de vulnerabilidad.

1.4.2. Objetivos específicos

- Construir una base de datos de entrenamiento para un modelo de reconocimiento de captcha mediante redes neuronales convolucionales.
- Implementar un modelo que permite el reconocimiento de captchas con redes neuronales convolucionales.
- Evaluar la exactitud del captcha mediante una métrica de error en un entorno controlado.

1.5. HIPÓTESIS DE LA INVESTIGACIÓN

1.5.1. Hipótesis general

El modelo para el reconocimiento de captcha utilizando redes neuronales convolucionales permite la evaluación de vulnerabilidad del captcha.

1.5.2. Hipótesis específicas

- Es posible construir una base de datos de entrenamiento para un modelo de reconocimiento de captcha mediante redes neuronales convolucionales.
- Es posible implementar un modelo que permite el reconocimiento de captchas con redes neuronales convolucionales.
- Es posible evaluar la exactitud del captcha mediante una métrica de error en un entorno.

1.6. VARIABLES

1.6.1. Variable independiente

- Modelo de reconocimiento de caracteres.

1.6.2. Variable dependiente

- Evaluación de vulnerabilidad de captchas.

1.7. OPERACIONALIZACIÓN DE VARIABLES

Tabla 1

Operacionalización de variables

VARIABLE	DIMENSION E INDICADOR	VALOR
Modelo de reconocimiento de caracteres	Método de aplicación	➢ Redes neuronales convolucionales ➢ Procesamiento de imágenes - Tesseract OCR
Evaluación de vulnerabilidad de un captcha	➢ Matriz de confusión ➢ F1-score	➢ Precisión ➢ Exactitud ➢ Especificidad ➢ Sensibilidad

CAPÍTULO II

REVISIÓN DE LITERATURA

En este capítulo contiene de dos partes. En la primera sección comprende los conceptos fundamentales que serán abordados a lo largo de esta presente tesis. En la segunda sección tiene la recopilación de antecedentes que dieron origen a esta presente investigación.

2.1. MARCO TEÓRICO

2.1.1. CAPTCHA

Un captcha es un sistema encargado de proteger los sitios web de los bots, generando y calificando pruebas que los humanos pueden pasar, pero no los programas informáticos. Por ejemplo, los humanos pueden leer un distorsionado o con ruido, pero las computadoras actuales no. (Hasan, 2016)

2.1.1.1. Captcha de texto

Este tipo de captcha es el más usado por ser fácil de implementar, está basado en un texto que se representa distorsionado conteniendo letras y dígitos que se distinguen entre mayúsculas y minúsculas tal como se muestra en la Figura 1.

Figura 1

Captcha de texto

Nota: Generado por el módulo Captcha en lenguaje Python.

2.1.2. Inteligencia artificial

Russell & Norvig, (2010) sostienen que la inteligencia artificial define como el estudio de agentes que reciben percepciones del entorno y realizan acciones. Cada agente implementa una función que mapea secuencias de percepciones a acciones, y se cubren diferentes formas de representar estas funciones, como agentes reactivos, planificadores en tiempo real y sistemas de toma de decisiones.

Méndez et al. (2008) menciona que la fenomenología de la inteligencia artificial abarca una gama de hechos relacionados con la neurología y la cognición, desde los niveles intracelular y neuronal hasta los mecanismos y organizaciones superpuestas que dan lugar a las funciones globales de percepción, memoria, lenguaje, toma de decisiones, emoción y acción lo que condujo al llamado comportamiento inteligente en los humanos. Además, se propone que el papel de la inteligencia artificial es dotar a la neurofisiología y la ciencia cognitiva de herramientas conceptuales y formales para avanzar en la "neurociencia computacional" para fortalecer la investigación en neurofisiología y ciencia cognitiva.

2.1.2.1. Visión por computador

La visión por computadora es un campo de investigación de rápido crecimiento enfocado en cómo las computadoras pueden obtener una comprensión del mundo visual similar a los humanos (Szeliski, 2011).

2.1.2.2. Aprendizaje automático

El aprendizaje automático (en inglés, Machine Learning) es un subcampo de la inteligencia artificial (IA) que se centra en el desarrollo de algoritmos y modelos que permiten a las computadoras aprender automáticamente a través de la experiencia y los datos. En lugar de ser programadas explícitamente para realizar una tarea, las máquinas utilizan datos para mejorar su rendimiento en esa tarea específica (Sandoval Serrano, 2018).

2.1.2.3. Aprendizaje profundo

El aprendizaje profundo (en inglés, Deep Learning) se puede definir como una rama especializada del aprendizaje automático que se enfoca en construir y entrenar redes neuronales profundas para realizar análisis de datos complejos y tareas de toma de decisiones. En el contexto de esta definición, Chollet destaca la importancia de la profundidad de las redes, refiriéndose a la existencia de múltiples capas interconectadas de neuronas artificiales. Estas capas permiten que el modelo aprenda gradualmente representaciones abstractas y jerárquicas de los datos, lo que le permite comprender y generalizar patrones complejos. A través de métodos como la retro propagación y el ajuste de peso, las redes neuronales profundas pueden adaptarse y mejorar continuamente su rendimiento para tareas específicas como la visión por computadora, el procesamiento del lenguaje natural, entre otras tareas (Chollet, 2017).

A diferencia de las técnicas tradicionales de aprendizaje automático que se basan en la extracción manual de características, el

aprendizaje profundo utiliza redes neuronales artificiales para aprender automáticamente características y patrones complejos en los datos. Una de las principales ventajas del aprendizaje profundo es su capacidad para aprender características de alto nivel a partir de datos de entrada de baja dimensión, esto significa que los modelos de aprendizaje profundo pueden aprender de datos sin procesar, como imágenes y audio, sin la necesidad de una extracción manual de funciones. Además, los modelos de aprendizaje profundo pueden aprender de grandes conjuntos de datos y capturar patrones complejos y sutiles en los datos (LeCun et al., 2015).

2.1.2.4. Visión artificial

La visión artificial es una rama de la inteligencia artificial donde se usan técnicas que permiten obtener, procesar y analizar cualquier tipo de información espacial obtenida a través de imágenes digitales o multimedia (Moralejo, R. O., & Storni, A., 2018).

2.1.2.5. Detección de objetos

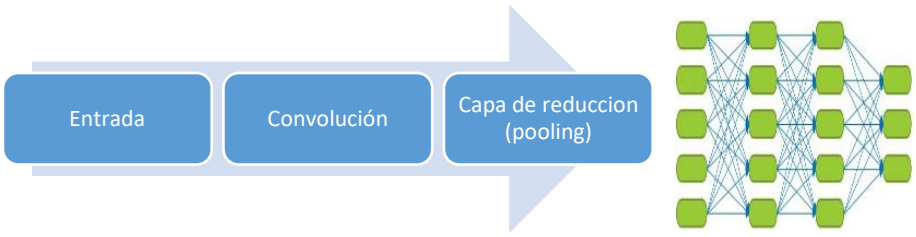
Antiguamente la visión artificial se enfocó solo a clasificar, pero hoy en día las nuevas técnicas son también detectoras. Esto implica que, además de determinar la categoría del objeto, podemos determinar su posición en relación a otros objetos y su tamaño, situándolo en un cuadro delimitador. Con el paso del tiempo, estos métodos de detección de objetos han progresado, y hoy en día podemos conseguir detecciones de objetos en imágenes en un periodo aproximado de 20 milisegundos, empleando equipos de costo relativamente asequible (Moralejo, R. O., & Storni, A., 2018).

2.1.2.6. Redes neuronales convolucionales

Según LeCun et al. (1989), una red neuronal convolucional es de una arquitectura especial de las redes neuronales artificiales basado en el algoritmo de aprendizaje profundo para el procesamiento del cerebro humano, diseñada específicamente para analizar y reconocer patrones en datos de tipo matricial, como imágenes y videos. La concepción de las redes neuronales convolucionales se basa en la idea fundamental de la convolución, una operación matemática que permite extraer características relevantes de los datos de entrada de manera eficiente (ver Figura 2).

Figura 2

Red neuronal convolucional



2.1.2.7. Convolución y su operación

Según Bosch et al. (2019), la convolución es una operación matemática que fusiona dos funciones para generar una tercera función (s) que muestra como una forma es medicada por la otra. Este proceso suele representarlo con un asterisco (∗):

$$s(t) = (f * g)(t) \int f(r)g(t-r)dr$$

Donde

$f(r)$: es el input o entrada

$g(t-r)$: es el filtro o kernel

$s(t)$: es el feature map o mapa de características

En caso de que utilicemos aplicaciones de aprendizaje automático, cuya entrada es un vector multidimensional de datos y el filtro (kernel) sea un vector multidimensional de parámetros podemos aplicar la siguiente formula:

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

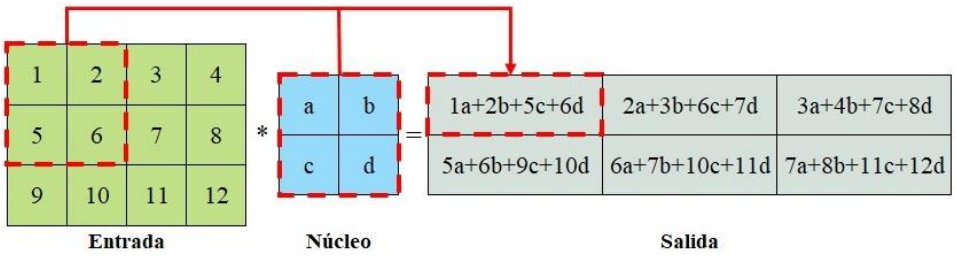
Por ejemplo, si empleamos el uso de imágenes (I) de dos dimensiones como imput es este caso usaremos un filtro o *kernel (K)* de dos dimensiones, podemos usar la siguiente formula:

$$s(i,j) = (I * K)(i,j) \sum_{m} \sum_{n} I(i-m, j-n)(m,n)$$

La Figura 3 muestra un ejemplo de convolución aplicada a un vector de datos bidimensional sin inversión del núcleo. La salida se visualiza solo donde el núcleo coincide con la imagen (nunca cruza los bordes). El cuadrado y las flechas muestran cómo se forma el elemento superior izquierdo aplicando el núcleo a la parte superior correspondiente de la región de entrada (Goodfellow et al., 2016).

Figura 3

Convolución 2D sin inversión del núcleo



Nota: Adaptado de sin correlación cruzada, Goodfellow et al., 2016.

2.1.3. Procesamiento multimedia

El procesamiento multimedia se refiere al proceso de tratar diferentes formatos como audio, video, texto, imágenes; mediante algún método o herramienta con el cual se pueda extraer características necesarias para poder analizarlos (Calizaya M. R., 2023).

2.1.3.1. Operaciones morfológicas

Las operaciones morfológicas son técnicas que se utilizan para modificar las formas de las imágenes a través de operaciones basadas en geometría. Estas operaciones ayudan a simplificar las imágenes mientras se mantienen las características principales. También se utilizan para tareas de procesamiento previo y posterior, como el filtrado morfológico, reducción de grosor y aumento de grosor.

2.1.3.1.1. Operador apertura

La operación de apertura (en inglés, opening) consiste en unir todas las traslaciones de un objeto estructural B que entran completamente dentro del objeto A. Al utilizar esta técnica se eliminan las áreas de un

objeto que no pueden ser contenidas por el objeto estructural, se suavizan los bordes y se eliminan los enlaces finos.

$$Apertura \rightarrow A \diamond B = (A \ominus B) \oplus B$$

2.1.3.1.2. Operación clausura

La operación de clausura (en inglés, closing) consiste en unir todas las traslaciones de un objeto estructural B que no se superponen con el objeto A. Esta técnica tiene como resultado suavizar los bordes de los objetos, pero al mismo tiempo engrosa los enlaces finos, rellenando "golfos" y agujeros más pequeños que el elemento estructural.

$$Clausura \rightarrow A \cdot B = (A \oplus B) \ominus B$$

2.1.3.1.3. Erosión

La operación de erosión (en inglés, erotion) tiene como objetivo afinar los objetos de una imagen mediante el uso de un elemento estructural. El elemento estructural se desplaza a través de la imagen y se determina si se encuentra completamente contenido en las áreas con valores de 1. Si esto es así, el resultado de la erosión será un valor de 1 en el punto de origen del elemento estructural

Matemáticamente.

$$Erosión \rightarrow A \ominus B = \{Z / B_Z \cap A^C \neq 0\}$$

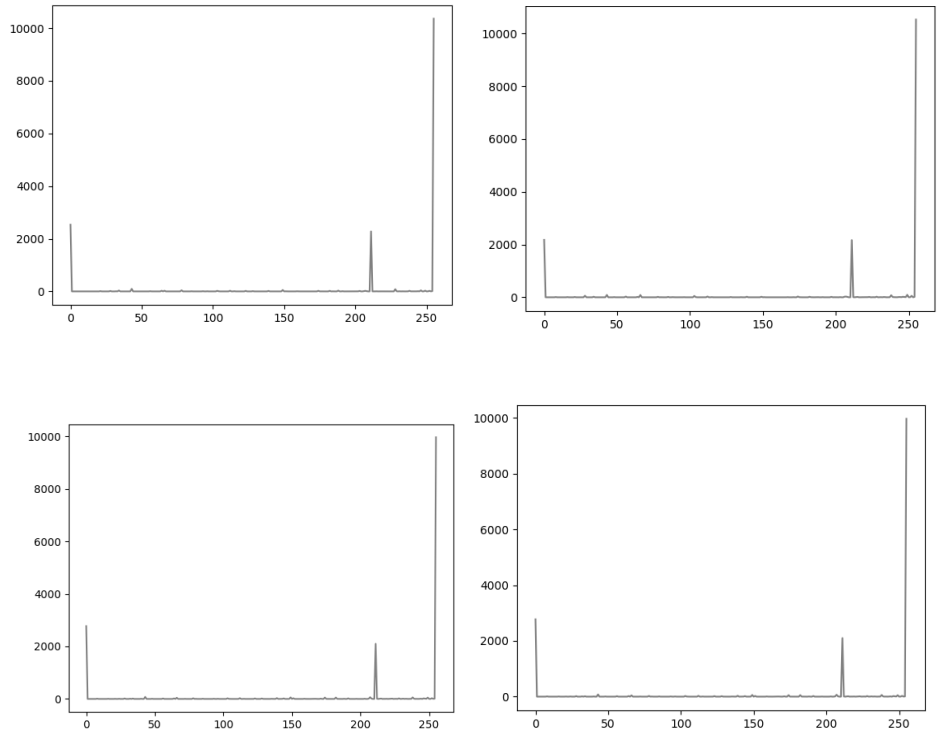
2.1.4. Histograma

Un histograma es una distribución que representa la frecuencia representada por los valores de intensidad en una imagen, independientemente del

modelo de color encontrado El modelo de color más fácil de entender es una imagen en escala de grises (Cuevas et al., 2018).

Figura 4

Histograma de varios captchas

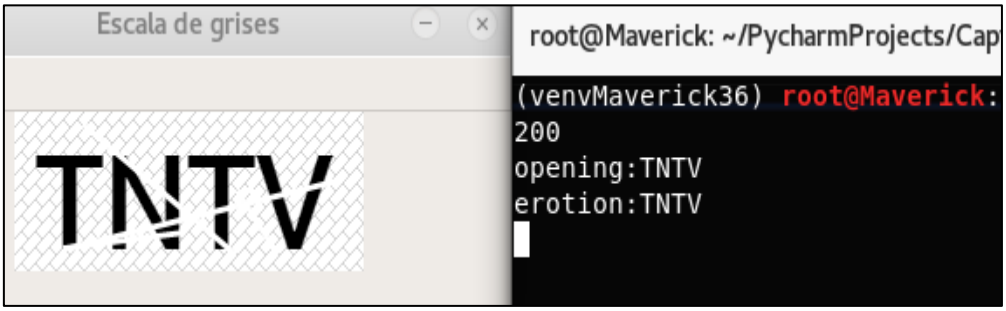


2.1.5. Reconocimiento óptico de caracteres

El reconocimiento óptico de caracteres (OCR, por sus siglas en inglés Optical Character Recognition) es un proceso en el que se transforma texto impreso o escrito manualmente en texto electrónico editable. Emplea algoritmos y técnicas de visión por computacional para detectar y distinguir patrones de texto en un documento o imagen escaneada, lo que posibilita efectuar búsquedas, modificaciones y otras características en el texto digitalizado, como se puede visualizar en la Figura 5 (Somoza Barreiro & Jiménez Gutiérrez, 2016).

Figura 5

Reconocimiento de caracteres



2.1.5.1. Procesamiento de imágenes

El procesamiento de imágenes es un término utilizado para referirse a las operaciones realizadas sobre un conjunto de datos de imágenes con el fin de mejorarlas de alguna manera, hacerlas más fáciles de interpretar o extraer algún tipo de información útil de ellas. Algunas de las principales operaciones de procesamiento de imágenes digitales incluyen escalado, codificación, extracción de características y reconocimiento de patrones (Malpartida, 2011).

En tales casos, es importante combinar el concepto de imagen digital con la función "$f(x, y)$" que tiene dos dimensiones o coordenadas espaciales "x" e "y" representadas de forma sencilla. imágenes. Por tanto, cuando se observa una imagen gris o en color, representa una descripción cuantitativa de las propiedades de la función. Por tanto, una imagen digital se refiere a una matriz en la que cada elemento corresponde a un valor de la función " $f(x, y)$" expresado en un número finito de bits, mientras que el índice de fila y la columna representa las posiciones "x" e "y". Cada elemento de la matriz se denomina píxel (elemento de imagen). Los

píxeles pueden ser escalares o vectores del mismo tamaño para toda la imagen, estos últimos pueden considerarse como compuestos por varios planos formados por píxeles escalares (Roncagliolo, 2007).

2.1.5.2. Umbralizacion

La umbralización es un método de segmentación sencillo y eficaz que facilita la clasificación de los pixeles de la imagen en escala de grises en dos grupos a partir de una ventana o un margen límite (Cortés et al., 2011).

2.1.5.3. Binarización

Antes de procesar una imagen, normalmente buscamos áreas en la imagen que creemos que son picos, o buscamos comportamiento en un conjunto de píxeles que nos permite determinar si pueden clasificarse como pertenecientes al mismo patrón o región. Esto implica convertir la imagen a escala de grises, donde a cada píxel se le asigna un valor entre 0 y 255 (Taquía-Gutiérrez, 2017).

2.1.5.4. Sustracción de fondo

Existen muchos algoritmos que existen para implementar un reconocimiento de objetos y este en caso de la sustracción de fondo, la mayoría de ellos siguen un diagrama de flujo simple definido por Cheung & Kamath (2004), que pasa por cuatro pasos generales:

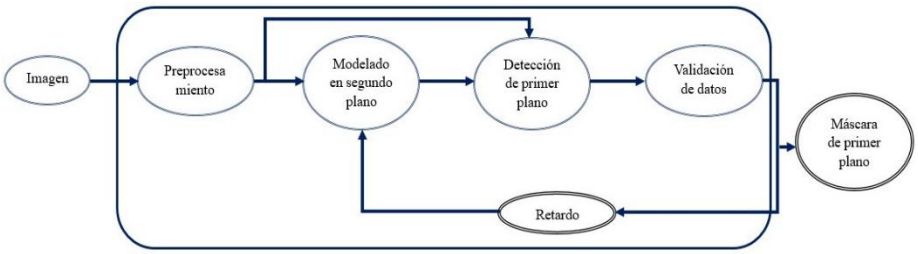
- Preprocesamiento: tareas simples de procesamiento de imágenes que convierten la entrada de video a un formato que se puede procesar en pasos posteriores.

- Modelado de fondo: También conocido como mantenimiento en fondo.
- Detección de primer plano: también conocida como sustracción de fondo.
- Validación de datos: También conocida como post - procesamiento, se utiliza para eliminar píxeles que no coinciden con objetos en movimiento.

Escobar Tafurt (2016) define que: "En una sustracción los píxeles que no cambian son considerados como ‘background’ y los píxeles que cambian son considerados como ‘foreground’ o u objetos en movimiento", como se muestra en la Figura 6 .

Figura6

Etapa de sustracción de fondo



Nota: Adaptado de Fernando & Tafurt, 2016.

2.1.6. Herramientas de desarrollo

2.1.6.1. Función Matplotlib

La función Matplotlib es un conjunto de funciones que permite que el uso de la librería era similar a la de MATLAB Estas funciones realizan diversas acciones en una figura, como la creación de figuras y áreas de

impresión, la impresión de líneas y etiquetas en las figuras, entre otras (Rivera Demanuel et al., 2022).

2.1.6.2. Matriz de confusión

Una matriz de confusión es una herramienta que muestra los resultados de una serie de pruebas en un problema de clasificación. En esta matriz, cada columna simboliza las instancias de una clase predicha, mientras que cada fila exhibe las instancias de una clase real. Los componentes de la matriz de la matriz de confusión establecen la cantidad de ejemplos de prueba que se alinean con la clase real de la fila y con la clase predicha de la columna. Un rendimiento óptimo se manifiesta en cifras elevadas a lo largo de la diagonal central y en valores reducidos, idealmente cero, en los componentes que se encuentran fuera de la misma (Witten et al., 2011).

2.1.6.3. Exactitud

La métrica exactitud (en inglés, accuracy) es una medida de evaluación que se utiliza comúnmente en problemas de clasificación. Se basa en contar el número de veces que la predicción del modelo coincide con la etiqueta real en los datos de entrenamiento y dividir este número entre el total de elementos en el conjunto de datos. Es una forma de medir cuán bien el modelo está clasificando los datos (Puig et al., 2019).

De forma matemática, la exactitud funciona de esta manera:

$$exactitud = \frac{VP + FP}{VP + VN + FP + FN}$$

Donde:

FP: Falsos positivos (False positives).

VP: Verdaderos positivos (True positives).

FN: Falsos negativos (False negatives).

VN: Verdaderos negativos (True negatives).

2.1.6.4. Exhaustividad

La exhaustividad (en inglés, recall) en aprendizaje automático es una métrica utilizada para medir la capacidad del modelo para detectar correctamente todos los casos positivos en un conjunto de datos de prueba. Es una medida de sensibilidad o verdadero positivo, y se calcula dividiendo el número de casos positivos correctamente identificados por el número total de casos reales positivos. Esto ayuda a medir qué tan bien el modelo es capaz de detectar todas las ocurrencias reales de un evento como se muestra en la Tabla 2. Es usado para medir la eficacia del modelo en problemas de clasificación binaria (Heras, 2020).

Tabla 2

Cuadro de exhaustividad

		Predicción	
		0	1
Realidad	0	VN	FP
	1	FN	VP

Nota: Adaptado de (Heras, 2020).

Como se verifica en la siguiente fórmula para calcular la exhaustividad es (verdaderos positivos) / (verdaderos positivos + falsos

negativos). La exhaustividad mide la capacidad de un modelo de clasificación para detectar correctamente los casos positivos en un conjunto de datos. Es decir, mide la proporción de casos positivos correctamente identificados en relación con el número total de casos positivos reales. Un valor de exhaustividad alto indica que el modelo es bueno en la detección de casos positivos, mientras que un valor bajo indica lo contrario.

$$Exhaustividad = \frac{VP}{vP + FN}$$

Donde:

FN: Falsos Negativos (False negatives).

VP: Verdaderos Positivos (True positives).

2.1.6.5. Precisión

La precisión es una métrica utilizada en aprendizaje automático para medir la exactitud de las predicciones de un modelo. Se calcula dividiendo el número de predicciones correctas entre el número total de predicciones realizadas. La precisión se utiliza a menudo en problemas de clasificación binaria, donde el objetivo es identificar correctamente si una muestra pertenece a una clase específica o no. Un modelo con una precisión alta indica que tiene un bajo nivel de falsos positivos (Heras, 2020).

Para calcular la precisión usaremos la siguiente formula:

$$precisión = \frac{VP}{VP + FP}$$

Donde:

FP: Falsos positivos (False positives).

VP: Verdaderos positivos (True positives)..

2.1.6.6. F1-score

La medida F1-score combina las métricas de precisión y exhaustividad (recall), mostrando diferencias en el rendimiento de un clasificador que no son evidentes solo con la precisión que es el número de verdaderos positivos dividido por el número de verdaderos positivos más falsos positivos y la exhaustividad es el número de verdaderos positivos dividido por el número de verdaderos positivos más falsos negativos. Es directamente proporcional al aumento de ambas medidas, por lo tanto, valores altos de F1-score indican que el algoritmo de clasificación tiene una mejor capacidad de predecir la clase positiva (Bekkar et al., 2013).

2.1.6.7. F – Measure

F-Measure es una métrica de evaluación utilizada en problemas de clasificación binaria que combina las métricas de precisión y exhaustividad . Fue propuesta por Davis & Goadrich (2006) en el artículo titulado "The Relationship between Precision-Recall and ROC Curves". La medida F1-score se calcula como la media armónica de la precisión y exhaustividad, donde un valor alto de F1-score indica un buen desempeño en ambas métricas (Van Rijsbergen, 1979).

$$F - measure = \frac{2 * exhaustividad * precisión}{exhaustividad * precisión}$$

Dónde precisión es la proporción de casos verdaderos positivos (True Positives) entre los casos verdaderos positivos y los casos falsos positivos (False Positives), y exhaustividad es la proporción de casos verdaderos positivos entre los casos verdaderos positivos y los casos falsos negativos (False Negatives).

2.1.6.8. OpenCV

OpenCV es una biblioteca de visión computacional, orientado al procesamiento y análisis de imágenes, implementado en C y C++, con soporte para diferentes sistemas operativos como Linux, Windows, Mac OS x. Su aplicación se destaca en áreas como imágenes médicas, inspección y seguridad, juegos y controles, robótica, etc, también incluye una biblioteca de aprendizaje automático, reconocimiento facial y realidad aumentada (Minguez, 2021).

2.1.7. Definición de términos básicos

2.1.7.1. Prueba de Turing

La prueba de Turing, propuesta por Alan Turing en 1950 en un artículo publicado en la revista Mind, es un método utilizado para evaluar la existencia de inteligencia en una máquina y sigue siendo uno de los mejores enfoques para los defensores de la Inteligencia Artificial (Turing, 2009).

2.1.7.2. Jupyter Notebook

Jupyter Notebook es una herramienta web que permite crear y compartir documentos computacionales. Su interfaz es intuitiva,

optimizada y enfocada en la presentación de documentos (Mendez et al., 2019).

2.1.7.3. Anaconda Navigator

Anaconda Navigator es una herramienta de escritorio que ayuda a organizar entornos de Python y paquetes de Anaconda de manera fácil y eficiente, proporciona una interfaz gráfica para facilitar la instalación de paquetes, lanzamiento de programas y manejo de entornos virtuales. También ofrece una plataforma para el uso de Jupyter Notebook y otras herramientas relacionadas con la ciencia de datos y el aprendizaje automático. En resumen, es una herramienta para la gestión y control de paquetes y entornos de desarrollo para Python y otros lenguajes (Rolon-Mérette et al., 2016).

2.1.7.4. Modelo

Un modelo de inteligencia artificial es una estructura computacional que puede aprender patrones complejos en datos utilizando algoritmos de aprendizaje automático y particularmente en el contexto de redes neuronales profundas (Goodfellow et al., 2016).

2.1.7.5. Librería Keras

La librería Keras es una biblioteca de código abierto escrita en Python, diseñada para acelerar el proceso de creación de redes neuronales. Se basa en el trabajo del desarrollador de Google, François Chollet, en el proyecto ONEIROS y fue lanzada por primera vez en marzo de 2015. Keras no funciona como un entorno de trabajo (framework) independiente,

sino como una interfaz fácil de usar (API) que permite acceder a diferentes entornos de aprendizaje automático y desarrollarlos. Entre los entornos compatibles con Keras se incluyen Theano, Microsoft Cognitive Toolkit y TensorFlow (Torres, 2020).

2.1.7.6. Librería TensorFlow

La librería TensorFlow es un sistema de código abierto para computación numérica desarrollado por Google Cloud. Es ampliamente utilizado en varios productos de Google Cloud para desarrollar algoritmos inteligentes. Google Cloud es un líder en campos como la Inteligencia Artificial, el aprendizaje automático y el Big Data, y TensorFlow surge de uno de sus proyectos en estas áreas, llamado Google Brain (Torres, 2020).

2.1.7.7. Python

Python es un lenguaje de programación de más alto nivel, puesto que las instrucciones en el código son más entendibles por un humano (Sergio Delgado Quintero, 2022).

Python permite reflejar de forma simple y elegante, las ideas en forma algorítmica, tiene una sintaxis sencilla y clara que permite escribir programas en menos tiempo, además Python tiene un modo interpretado lo que permiten, que ofrece retroalimentación inmediata, lo que permite probar nuevas ideas casi al instante (Dawson, 2010).

2.1.7.8. Librería Pandas

La librería Pandas es un paquete open-source que nos proporciona una forma sencilla y potente de trabajar con estructuras de datos a través de múltiples herramientas para su análisis (Delgado Quintero, 2022)..

2.1.7.9. Librería Numpy

La librería Numpy es una herramienta esencial para la manipulación y el análisis de datos en Python, proporcionando estructuras de datos eficientes, algoritmos optimizados y funciones para realizar operaciones numéricas rápidas en arreglos (McKinney, 2022).

2.1.7.10. Tabla de datos

Una tabla de datos (en inglés, DataFrame) es un tipo de estructura de datos que se presenta en forma de tabla en Python y son una de las herramientas principales del paquete pandas para el análisis de datos. Permiten almacenar y procesar información en una tabla similar a una hoja de cálculo, con filas y columnas identificadas. Son muy útiles para realizar tareas de limpieza, transformación y análisis de datos (Wu, 2020).

2.1.7.11.Entorno virtual Virtualenv

Virtualenv es una herramienta utilizada para el desarrollo de software en el lenguaje de programación Python. Le permite crear un entorno virtual independiente en el que puede instalar paquetes y dependencias específicas para su proyecto sin afectar el sistema Python global. Estos entornos virtuales actúan como contenedores aislados que encapsulan las bibliotecas y versiones de Python utilizadas en un proyecto

en particular, lo que facilita la gestión de dependencias y garantiza la reproducibilidad del entorno de desarrollo en diferentes sistemas (Reitz & Schlusser, 2016).

2.1.7.12. Tesseract OCRts

Tesseract es un programa libre que permite reconocer caracteres en imágenes. Es compatible con varios sistemas operativos y su desarrollo es patrocinado por Google desde 2006. Está liberado bajo la licencia Apache 2.02 y ha sido considerado como uno de los mejores programas de reconocimiento óptico de caracteres de código abierto en 2006 (Smith, 2007).

2.2. ANTECEDENTES

Aguilar Fauta & others, (2021) en la tesis de investigación propone un método en el cual lo primero que se hace es segmentar, limpiar los caracteres mediante procesamiento de imágenes, adicionalmente a esto se utiliza una arquitectura en redes neuronales convolucionales y en este modelo se prueba diferentes configuraciones de hiper parámetros buscando los mejores resultados, obteniendo como resultado una precisión del 97.10% y con esto demostrando que con estos son vulnerables.

Lin et al., (2018) los investigadores indican que la mayoría de trabajos presentado anteriormente están basados en caracteres digitales y en inglés estos investigadores presentan un trabajo con caracteres chinos debido a su dificultad en el reconocimiento automático para su investigación proponen una arquitectura basada en redes neuronales convolucionales e indican que esta arquitectura es superior a LENET. Los resultados son que el modelo propuesto logra más del 95 % de precisión para un solo carácter chino y un 84 % de precisión para tres tipos de CAPTCHA de caracteres chinos con cuatro

caracteres chinos y que es muy superior a algún software de reconocimiento de CAPTCHA y reconocimiento óptico de caracteres.

Garcia Serrano, Edgar D. (2018) en su tesis indicaba que un captcha es un programa que niega o permite el acceso a servicios mediante la generación y calificación de pruebas que las personas humanas pueden pasar y que los problemas de computadora que en ese tiempo no podían realizar, muchas de las pruebas que pasaron basado en reconocimiento de texto han sido vulnerados por técnicas de reconocimiento óptico de caracteres (OCR, por las siglas en ingles), mientras que aquellos que estaban basados en imágenes son vulnerables a ataques de aprendizaje automático (machine learning). el investigador presenta sietes modelos de captchas para probar algunas habilidades cognitivas que en teoría van más allá de las capacidades de los agentes artificiales, de las cuales cuyos resultados en las pruebas, pudo ver que los agentes artificiales pudieron emular la habilidad humana para reconocer imágenes como fotografías, pero no eran suficientemente para competir con la habilidad humana.

Zhou et al. (2013) los investigadores proponen un nuevo modelo de red neuronal que llaman ADN (Active Deep Network) y hacen referencia a la dificultad de realizar una investigación con insuficientes datos etiquetado, el cual comparan con métodos de clasificación actuales como aprendizaje espectral (en inglés, spectral learning), SVM, aprendizaje activo, minería de la clasificación fácil, redes de creencias profundas y autocodificadores recursivos. Aprendizaje espectral, TSVM obteniendo mejores resultados.

2.2.1. Artículos de investigación

Walia & odugoudar (2023) en su investigación comprendía que muchos sitios web mejoraban su seguridad y prevenían ataques maliciosos en línea

implementando CAPTCHA (Prueba de Turing pública completamente automatizada para distinguir computadoras de humanos), un tipo de prueba determinista para determinar si el usuario final es un humano o un robot. El tipo de CAPTCHA más común es textual y está diseñado para que los humanos reconozcan fácilmente, aunque no existe una solución para máquinas o robots. Sin embargo, a medida que avanza la tecnología de aprendizaje profundo, resulta más fácil desarrollar modelos de redes neuronales convolucionales (CNN) para predecir CAPTCHA de texto. El propósito de este estudio es investigar las debilidades y vulnerabilidades en el sistema de generación de CAPTCHA para desarrollar CAPTCHA más confiables. Para ello se creó CapNet, una red neuronal convolucional. El enfoque más sencillo es ampliar el conjunto de entrenamiento inicial con datos adicionales. Esto es para ver si podemos acceder a todos los estilos de fuente adicionales que se pueden usar para crear un CAPTCHA sin revisarlos específicamente en el tutorial. Además, la optimización de hiper parámetros se puede utilizar para intentar optimizar una única arquitectura (esto también requiere más potencia informática). Sin embargo, debido a que el espacio de parámetros de CNN es muy grande, otros enfoques, como el metamodelo, pueden ser más factibles.

Villares-Jimenez Julio & Quinatoa-Medina (2023) los investigadores pusieron en conocimiento el tratado de pruebas de respuesta a desafíos guiadas por máquina para determinar si el usuario es un ser humano o un programa automatizado (bot). Los ataques de bots maliciosos son uno de los problemas más comunes en los sistemas en línea. Para combatir estos ataques, se puede implementar un sistema CAPTCHA. En este contexto, el artículo tiene como objetivo automatizar la tarea de encontrar y analizar vulnerabilidades en sistemas

web utilizando herramientas de automatización de procesos robóticos (RPA) y técnicas de visión artificial. Como parte del desarrollo del programa automático (bot), se utilizó la herramienta UiPath Community Edition y la API de Google Cloud Platform para métodos de visión artificial. El propósito de esta propuesta es demostrar que los sistemas CAPTCHA, especialmente el tipo hCAPTCHA, se pueden eludir utilizando técnicas de visión artificial combinadas con procesos de bot automatizados. Por lo tanto, el rendimiento del bot desarrollado con el sistema CAPTCHA roto se evaluó sistemáticamente basándose en la imagen de la página de prueba.

Dankwa & Yang (2021) en su investigación proponía que los profesionales de la ciberseguridad generan una prueba de Turing pública completamente automatizada para diferenciar las computadoras de los humanos (CAPTCHA) como una forma de mecanismo de seguridad en las aplicaciones de sitios web, con el fin de diferenciar entre usuarios finales humanos y robots de máquinas, el CAPTCHA basado en texto es el más utilizado. Sin embargo, con la evolución del aprendizaje profundo, ha sido tan dramático que es posible descifrar tareas que antes se pensaba que no eran fáciles de abordar mediante computadoras y que se usaban como CAPTCHA para evitar el spam. El flujo de trabajo de ruptura de CAPTCHA es una combinación de esfuerzos, enfoques y el desarrollo del modelo de red neuronal convolucional (CNN) computacionalmente eficiente que intenta aumentar la precisión. Se ha evaluado y comparado el rendimiento de nuestro modelo propuesto con el de ajustar otras arquitecturas populares de reconocimiento de imágenes CNN en el conjunto de datos de imágenes CAPTCHA generado. En tiempo real, nuestro modelo propuesto utilizó menos

tiempo para descifrar los CAPTCHA basados en texto con una precisión de más del 99% en el conjunto de datos de prueba.

Noury & Rezaei (2020) presentaron una investigación que describió el cómo descifrar un CAPTCHA que es una prueba centrada en el ser humano que distingue a los operadores humanos de los programas automáticos (bots), atacantes u otros actores informáticos que intentan imitar la inteligencia humana, intuitivos utilizando una solución automatizada basada en aprendizaje profundo. El propósito de este estudio fue investigar las vulnerabilidades y vulnerabilidades de distintos sistemas de generación de CAPTCHA, así que desarrollaron CAPTCHA más potentes sin el riesgo de prueba y error manual. Para lograr este objetivo, desarrollamos una red neuronal convolucional llamada DeepCAPTCHA. La plataforma propuesta permite el estudio de CAPTCHA tanto numéricos como alfanuméricos. Para entrenar y desarrollar un modelo eficiente, creamos un conjunto de datos de 500.000 CAPTCHA para entrenar nuestro modelo, presentaron su propio modelo de red neuronal profunda, analizaron los problemas que existen y cómo resolverlos. La precisión del crackeo de la red neuronal arroja puntuaciones altas de 98,94% y 98,31%, respectivamente, para conjuntos de datos de prueba numéricos y alfanuméricos. Como resultado de esta investigación, hemos identificado métodos efectivos para mejorar la seguridad de los CAPTCHA basados en el análisis de rendimiento realizado utilizando el modelo Deep-CAPTCHA.

Wang et al. (2019) los investigadores apuntaron a los problemas de baja eficiencia y poca precisión de los métodos tradicionales de reconocimiento CAPTCHA, hemos propuesto una forma más eficiente basada en una red neuronal convolucional profunda (CNN). La red convolucional densa (DenseNet) ha

demostrado un excelente rendimiento de clasificación que adopta una conexión entre capas. No solo alivia eficazmente el problema del gradiente de fuga, sino que también reduce drásticamente la cantidad de parámetros. Sin embargo, también ha provocado un gran consumo de memoria. Por eso mejoramos y construimos una nueva DenseNet para el reconocimiento CAPTCHA. Los experimentos muestran que la nueva red no sólo mantiene las principales ventajas de rendimiento de DenseNets sino que también reduce efectivamente el consumo de memoria. Además, la precisión del reconocimiento de CAPTCHA con ruido de fondo y adherencia de caracteres es superior al 99,9%.

Hu et al. (2018) los investigadores proponen un método basado en redes neuronales convolucionales (CNN) para identificar los captcha para evitar el uso de técnicas de procesamiento de imágenes tradicionales (operadores de procesamiento de imágenes) y los autores hacen énfasis en que todos los caracteres pueden ser reconocidos sin segmentar las imágenes, se introduce la tasa de aprendizaje adaptativo para acelerar la tasa de convergencia del modelo, y las conclusiones son que el modelo es bastante bueno ya que la precisión alcanza el 95% y tiene por trabajas futuros realizar el entrenamiento con caracteres chinos.

Y. Wang & Lu (2018) en su investigación indicaron que los CAPTCHA (prueba pública de Turing completamente automatizada para diferenciar las computadoras de los humanos) se puede utilizar para proteger los datos de los robots automáticos. De este modo se diseñan innumerables tipos de CAPTCHA, mientras que nosotros utilizamos con mayor frecuencia esquemas basados en texto debido a su mayor comodidad y facilidad de uso. En la investigación, construimos un sistema completo para derrotar los CAPTCHA y lograr un rendimiento de última generación. En detalle, presentaron el algoritmo auto

adaptativo para segmentar diferentes tipos de caracteres de manera óptima y luego utilizamos tanto los métodos existentes como nuestra propia red neuronal convolucional construida como clasificador adicional.

Stark et al. (2015) realizaron la investigación donde los investigadores indica que la potencia de las redes neuronales convolucionales es aprovechable siempre y cuando el conjunto de datos es muy grande sin embargo proponen un método según el cual a partir de pocos datos etiquetados se autogeneren un conjunto de datos Captcha que llaman profundidad activa. También indican en sus resultados que mejoran el desempeño de la red con las muestras correctamente clasificadas pero inciertas.

Yamaguchi et al. (2014) en su investigación describía que muchas personas con discapacidad visual se quejan de la mala accesibilidad de los sistemas CAPTCHA convencionales porque la prueba de estilo de audio es demasiado difícil para los humanos. En este estudio demostramos la inseguridad de este tipo de sistema CAPTCHA. Demostramos que nuestro programa de resolución puede superar el CAPTCHA con una tasa de éxito superior al 99%. Además, proponemos un nuevo sistema de estilo verbal para reemplazar el CAPTCHA basado en cuestionarios. Nuestro sistema sintetiza varias oraciones, que tienen diferentes grados de naturalidad en términos de su significado contextual, a partir de un conjunto de documentos fuente utilizando una cadena de Márkov de orden flexible. Esta prueba se implementa en estilo verbal, lo que significa que es universalmente adecuada para cualquier tipo de canal de percepción. Implementaron el esquema propuesto y analizamos su seguridad en base a experimentos.

Ling-Si & Yi-Chun, (2012) en su investigación explicaba que los CAPTCHA (Prueba de Turing pública completamente automatizada para diferenciar las computadoras de los humanos) se usa más que antes y se convierte en la parte común del sistema de inicio de sesión de sitios web actual. Sin embargo, la implementación de CAPTCHA es complicada y arriesgada sin un diseño deliberado. Nuestro objetivo es un sitio web del banco principal de China, demostraron que, con algunos métodos especializados, el esquema CAPTCHA en su sitio web se puede descifrar fácilmente, cuyos resultados después de varios experimentos, ajustando los parámetros de la red BP, encontramos que la tasa de identificación ideal es del 75%. Sin embargo, la tasa de reconocimiento sigue aumentando. Analizaron todos los casos de falla de nuestro ataque tanto en el conjunto de muestra como en de prueba. Encontraron varios aspectos que podemos mejorar en su propio sistema, dado que todas las muestras de las bibliotecas de muestras de formación no son muestras de bibliotecas de formación estándar. En nuestro ataque, utilizamos la red neuronal BP como clasificador para lograr el reconocimiento. Finalmente, damos algunos consejos a los diseñadores de CAPTCHA para que revisen la seguridad de nuestra implementación de CAPTCHA en el futuro.

Zhu et al. (2010) presentaron los investigadores estudiaron sistemáticamente el diseño de CAPTCHA de reconocimiento de imágenes (IRC). Primero revisamos y examinamos todos los esquemas de IRC existentes y evaluamos cada esquema con respecto a los requisitos prácticos de las aplicaciones CAPTCHA, particularmente en aplicaciones de la vida real a gran escala como Gmail y Hotmail. Luego presentamos un análisis de seguridad de los esquemas representativos que hemos identificado. Para los esquemas que

permanecen intactos, presentamos nuestros novedosos ataques, en el cual proporcionaron un marco simple pero novedoso para guiar el diseño de IRC robustos, un IRC innovador llamado Cortcha que es escalable para cumplir con los requisitos de aplicaciones a gran escala. Se basa en reconocer objetos explotando el contexto circundante, una tarea que los humanos pueden realizar bien, pero las computadoras no. Se puede utilizar una cantidad infinita de tipos de objetos para generar desafíos, que pueden desactivar efectivamente el proceso de aprendizaje en ataques de aprendizaje automático. Cortcha no requiere que las imágenes de su base de datos de imágenes estén etiquetadas. La recopilación de imágenes y la generación de CAPTCHA se pueden automatizar completamente. Cuyos estudios de usabilidad indican que, en comparación con el CAPTCHA de texto de Google, Cortcha permite una tasa de precisión humana ligeramente mayor, pero, en promedio, lleva más tiempo resolver un desafío.

CAPÍTULO III

MATERIALES Y MÉTODOS

3.1. TIPO DE INVESTIGACIÓN

La presente investigación tiene por finalidad de establecer el enfoque cuantitativo, ya que va enmarcada en la utilización de métodos y técnicas cuantitativas, con ello la recopilación y el análisis de datos, para responder preguntas de investigación y probar hipótesis pre formuladas. También se basa en la medición de variables e instrumentos de investigación utilizando estadística descriptiva e inferencial (Ñaupas Paitán, 2018).

Adicionalmente según Mousalli-Kayat (2015) la investigación proyectiva está relacionada con la elaboración de un modelo o propuesta para solucionar cualquier problema, planteando un diseño y proporcionar lineamientos para acciones inmediatas.

3.2. TÉCNICAS E INSTRUMENTOS DE RECOLECCIÓN DE DATOS

3.2.1. Técnicas

En esta investigación solo se basa en la utilización de la técnica de la observación directa, es decir que los procedimientos se realizan de manera simultánea, también se realiza la búsqueda de imágenes de captchas, las técnicas en el presente trabajo son las herramientas y procedimientos específicos empleados dentro de los métodos más amplios para ejecutar tareas concretas, para luego realizar estas técnicas que se detallan a continuación.

- Preprocesamiento de datos: estas técnicas incluyen el uso de bibliotecas específicas como OpenCV para el preprocesamiento de imágenes, la creación de clases generadoras de datos para la gestión eficiente de la

alimentación de datos durante el entrenamiento y cuyas características de cada imagen tiene una dimensión de 200 x 80 pixeles, donde se realiza el proceso de depuración del conjunto de datos, eliminando archivos de calidad baja.

- Implementación de un modelo de Inteligencia Artificial: donde se realiza la utilización de algoritmos de aprendizaje profundo y de aprendizaje automático, como redes neuronales convolucionales y LTSM. Además, técnicas como la decodificación para interpretar las salidas del modelo en tareas de reconocimiento óptico de caracteres, optimizando el rendimiento y la precisión del modelo en tareas específicas.
- Evaluación del modelo: donde tiene por finalidad la utilización de las métricas seleccionadas para esta investigación las cuales son la exactitud, error, precisión, sensibilidad, especificidad y F1-score.

3.2.2. Métodos

Los métodos en inteligencia artificial y el procesamiento de datos representan enfoques sistemáticos y estratégicos diseñados para abordar problemas complejos y alcanzar objetivos específicos. En el contexto del desarrollo de modelos de inteligencia artificial, los métodos incluyen la preparación y manipulación de datos, la selección y división de datos en conjuntos de entrenamiento y validación, la aplicación de técnicas de evaluación que se calculan de la matriz de confusión como precisión y exactitud. Estos métodos proporcionan un marco estructurado que guía la investigación y el desarrollo del presente trabajo.

3.2.3. Instrumentos

En primer lugar, el instrumento principal se tomó una base de datos creada para la implementación de los modelos con la que esta investigación está basada, en segundo lugar, se tomó en cuenta las guías de instrucciones de redes neuronales convolucionales y OpenCV como también las métricas para su evaluación todo ello implementado con librerías Python.

3.3. MÉTRICAS PARA LA EVALUACIÓN DEL MODELO

La matriz de confusión resumen en una sola tabla el número de predicciones correctas e incorrectas que realiza un modelo de CNN sobre un conjunto de datos, con filas que representan las clases reales y columnas las clases predichas (Tharwat, 2021).

Además de visualizar los aciertos y errores, la matriz de confusión permite derivar métricas como precisión, exhaustividad y F1-score para diferentes clases, entregando una evaluación comprehensive (Powers, 2020), como se visualiza en la Tabla 4:

Tabla 3

Matriz de confusión para detección de imágenes

		Valor Actual obtenido por experimento	
		Negativo	**Positivo**
Valor predicho por prueba	Negativo	Verdadero negativo	Falso negativo
	Positivo	Falso positivo	Verdaderos positivos

Nota: Adaptado de descripción de un matriz de confusión, por Mokhtari et al., 2021. https://arxiv.org/abs/2107.01031

3.4. POBLACIÓN Y MUESTRA

3.4.1. Población

La población está conformada por todas las imágenes de captchas que fueron generados por la librería de imágenes Gregwar Captcha de manera que son mostradas de manera aleatoria e infinitamente.

3.4.2. Muestra

Se ha empleado como muestra la base de datos en la Tabla 5. Se está empleando el muestreo probabilístico, puesto que implica que cada captcha en el conjunto de la población tiene una oportunidad de ser incluidos en la muestra según la complejidad del captcha, la frecuencia con la que aparece en el conjunto de datos o cualquier otro criterio relevante para el problema en cuestión. El muestreo aleatorio simple es una forma específica de muestreo probabilístico en

la que los elementos se seleccionan directamente según ciertos criterios predefinidos (Sánchez Carlessi et al., 2018).

Tabla 4

Caracteres de la base de datos

Descripción	Cantidad	Detalle
Número de carácteres únicos en la base de datos	26	['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
Número máximo de caracteres en los captchas	4	
Número total de muestras en base de datos	1017	

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

4.1. RESULTADOS

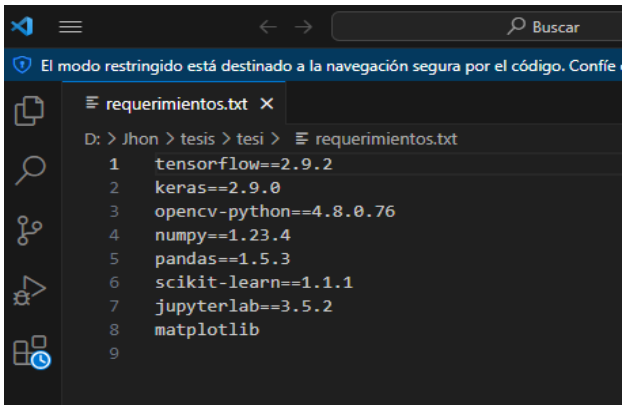
4.1.1. Construir una base de datos de entrenamiento para un modelo de reconocimiento de captcha mediante redes neuronales convolucionales

Para comenzar con la creación de la base de datos se procede a la carga los paquetes y si es necesario descargarlo e instalarlo en la máquina virtual de Windows Virtualenv por su rapidez y fluidez en el entorno de trabajo, pero también se puede trabajar en el software Anaconda Navigator además de tener instalado la versión 3.8.x de paquete de interprete de lenguaje Python que viene con varias optimizaciones de rendimiento, como llamadas de método más rápidas y una mejor gestión de la memoria.

En la Figura 7 se muestra la versión requerida para el funcionamiento de nuestros modelos, en el cual se visualiza que el paquete JupyterLab tiene la versión 3.6.3 y el paquete Jupyter Notebook que debe tener la versión 6.5.4.

Figura 7

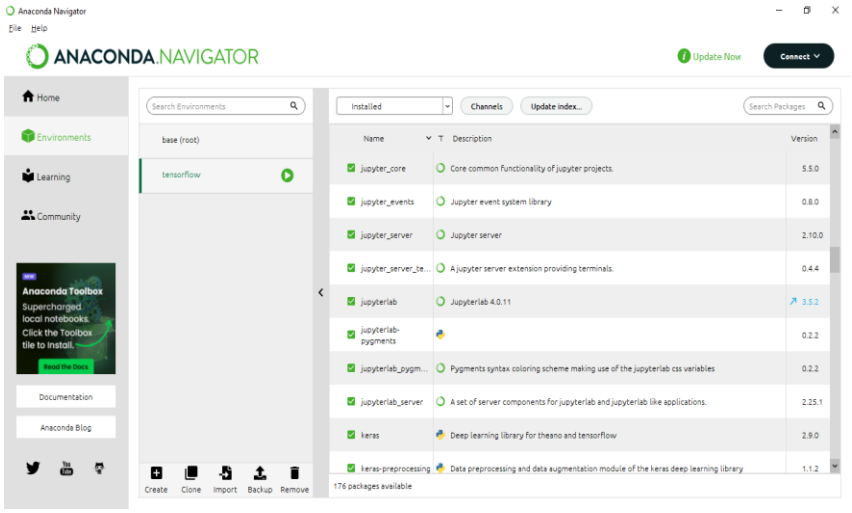
Versiones de paquetes



En la Figura 8 se visualiza los paquetes instalados por defecto en la máquina virtual llamada “tensorflow” que se creó mediante terminal con comandos de Python para la realización de nuestro trabajo.

Figura 8

Interfaz de Anaconda Navigator

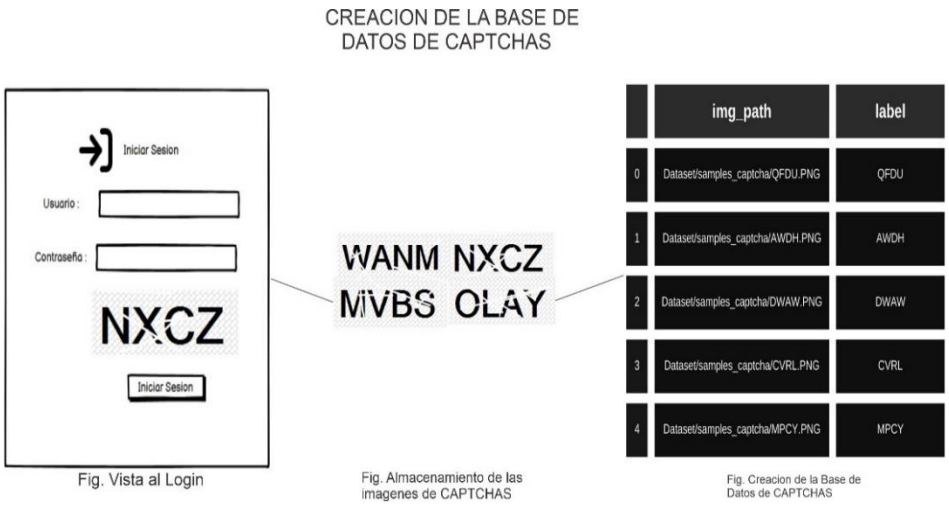


4.1.1.1. Creación de la base de datos de captchas

Para la elaboración de la base de datos de captchas utilizando la biblioteca Gregwar Captcha, donde genera imágenes aleatorias de captcha junto con sus etiquetas correspondientes. El proceso involucra la generación de imágenes a partir de caracteres al azar mediante la función de la librería, almacenándolas en un directorio específico, mientras que los textos de los captchas se registran en un archivo de etiquetas. Esta base de datos puede ser empleado para entrenar modelos de reconocimiento de texto y validación de imágenes CAPTCHA.

Figura 9

Proceso de creación de la base de datos



4.1.1.2. Construcción de la tabla de datos

Luego del etiquetado manual las muestras se han alojado en directorios desde los cuales han sido leídos por un algoritmo que se ha desarrollado para la construcción de una tabla de datos (en inglés, dataframe) con la dirección y la etiqueta de las letras, se tienen un total de 1017 muestras que posteriormente serán para el entrenamiento y la evaluación como se demuestra en la en la Tabla 6 y en la Figura 9.

Tabla 5

Tabla de datos construido

	Img_path	Label
0	Dataset/Samples_captcha/QFDU.png	QFDU
1	Dataset/Samples_captcha/AWDH.png	AWDH
2	Dataset/Samples_captcha/DWAW.png	DWAW
3	Dataset/Samples_captcha/CVRL.png	CVRL
4	Dataset/Samples_captcha/MPCY.png	MPCY

4.1.1.3. Reconocimiento de CAPTCHA mediante procesamiento de imágenes

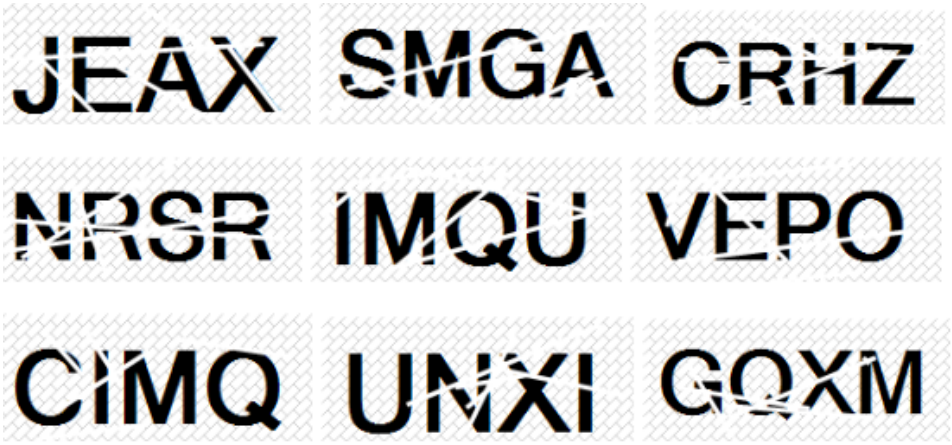
Se usa la técnica de conservación a escala de grises con el objetivo de reducir el número de colores, simplificar el procesamiento y reducir el tamaño.

4.1.1.4. Preprocesamiento

Con el fin de preprocesar las imágenes generadas del captcha se almacenaron 20 captchas de prueba recolectadas de la librería que se muestra en la Figura 10.

Figura 10

Ejemplos CAPTCHAS

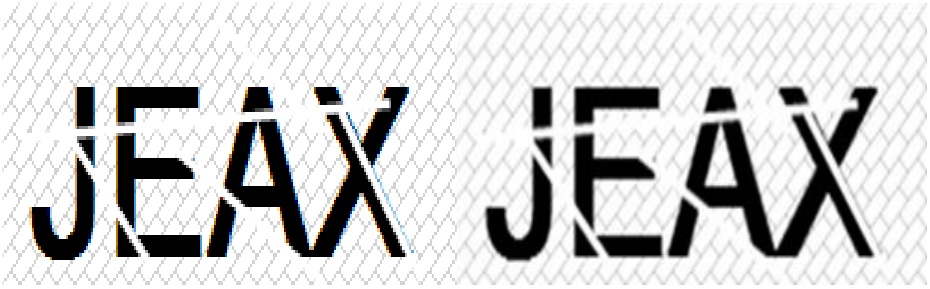


4.1.1.5. Histograma y filtrado por umbral

La conversión a escala de grises se realizó mediante OpenCV el procesamiento es asignar un valor de gris a cada pixel de la imagen en función de su valor de color original. En la Figura 11 se puede observar el captcha antes y después de la conversión a escala de grises.

Figura 11

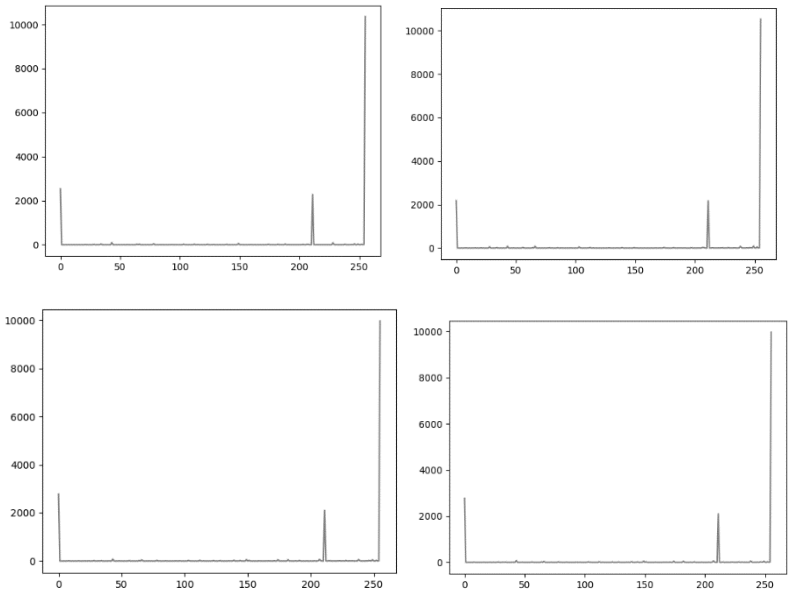
Captcha convertido



A continuación, se puede visualizar el histograma de los CAPTCHAS en la Figura 12.

Figura 12

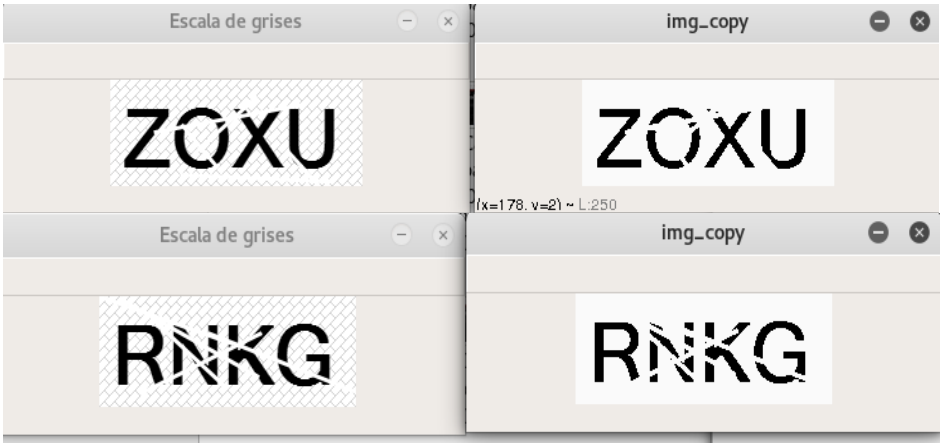
Histograma de varios captchas



Usando la herramienta Thresholding que se segmenta la imagen en dos regiones, una de fondo y otra de objeto mediante la comparación de los niveles de intensidad de cada píxel con un valor umbral de 175 en la Figura 13.

Figura 13

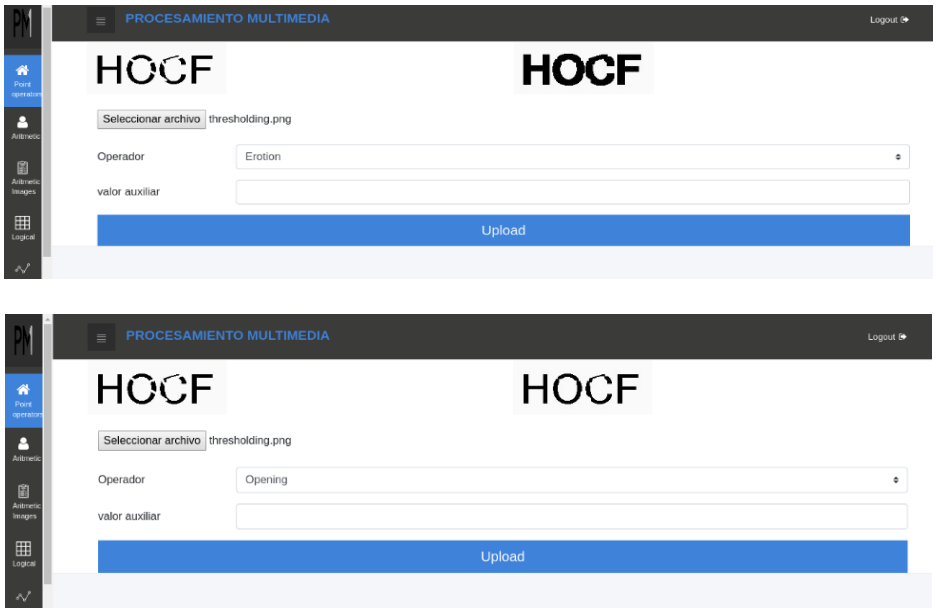
Aplicación de Thresholding



Para tener una tasa mayor de éxito en el reconocimiento óptico de caracteres se vio por conveniente completar los espacios en blanco se probaron distintos operadores morfológicos de los cuales se escogieron erosión y apertura de las cuales se puede visualizar en la Figura 14.

Figura 14

Operadores morfológicos Erosión y Apertura.

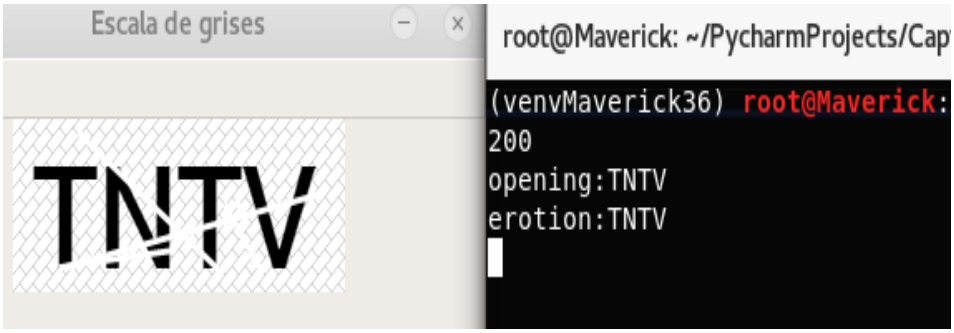


4.1.1.6. Segmentación y reconocimiento

Se divide la imagen en segmentos individuales haciendo uso de Tesseract, y se puede ver el resultado de la segmentación en la Figura 15.

Figura 15

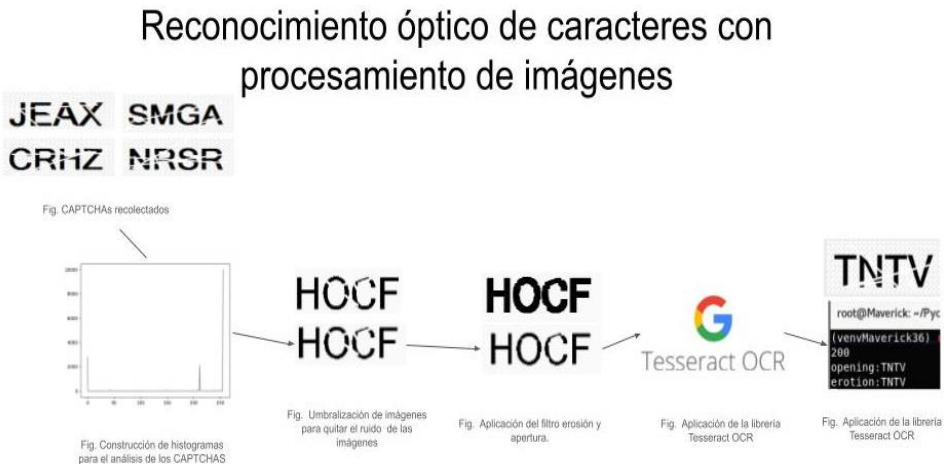
Reconocimiento de caracteres con Tesseract



En este caso, se evaluó esta propuesta con el procesamiento de imágenes con la herramienta Tesseract OCR como se puede ver en la Figura 16.

Figura 16

Propuesta de reconocimiento óptico de caracteres con procesamiento de imágenes



4.1.2. Implementar un modelo que permite el reconocimiento de captchas con redes neuronales convolucionales

4.1.2.1. Preprocesamiento y normalización de los datos

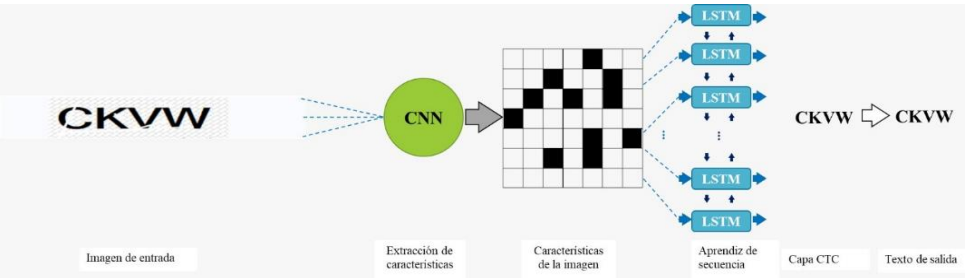
Uno de los procesos más habituales en los proyectos que conllevan análisis de datos es el preprocesamiento, donde se eliminan, imputan valores nulos o atípicos que puedan afectar el desempeño de los resultados del modelo. Otro proceso importante es la normalización de los datos, generalmente estos datos se convierten en valores entre 0 y 1 para evitar la redundancia de los mismos.

4.1.2.2. Arquitectura propuesta

Para el presente trabajo se utilizó la arquitectura llamada red neuronal convolucional CNN.

Figura 17

Arquitectura de una red neuronal convolucional



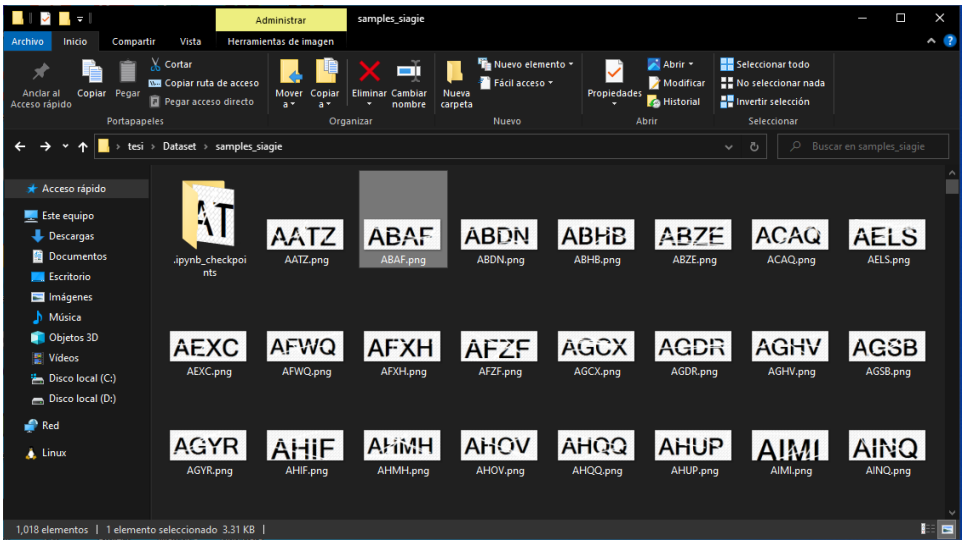
4.1.2.3. Entrenamiento

Una vez se ha construido la base de datos de captchas como se muestra en la Figura 18, estas muestras ingresan a los modelos de entrenamiento. Para el presente trabajo de investigación se entrenó un modelo basado en una red neuronal convolucional (CNN), ya que la

literatura ha demostrado que tienen mejores resultados en problemas de visión artificial como es el trabajo del reconocimiento óptico de caracteres.

Figura18

Base de datos creado.



Luego se importa las librerías de paquetes en el lenguaje de programación Python para el entrenamiento y lectura de datos como se muestra en la Figura 19.

Figura 19

Importación de paquetes en Jupyter Notebook

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from pathlib import Path
from collections import Counter
from sklearn.model_selection import train_test_split

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
print("Tensorflow version: ", tf.__version__)

seed = 1234
np.random.seed(seed)
tf.random.set_seed(seed)
```

```
Tensorflow version:  2.9.2
```

Para empezar a entrenar nuestro modelo, dividiremos la base de datos en conjuntos de datos de entrenamiento (training_data) y de validación (validation_data). El argumento test_size en train_test_split indica la proporción de datos que se reservan para el conjunto de validación (en este caso, el 10% de los datos). Luego, se llama a la función generate_arrays para generar un arreglo de imágenes y un arreglo de etiquetas para los conjuntos de datos de entrenamiento y validación. La función generate_arrays carga las imágenes, las redimensiona y las normaliza para que los valores de píxeles estén entre 0 y 1, y las etiquetas se codifican como enteros.

Figura 20

Código fuente del algoritmo de entrenamiento

```
training_data, validation_data = train_test_split(dataset, test_size=0.1, random_state=seed)
training_data = training_data.reset_index(drop=True)
validation_data = validation_data.reset_index(drop=True)

print("Numero de muestras para entrenamiento", len(training_data))
print("Numero de muestras para validación", len(validation_data))

char_to_labels = {char:idx for idx, char in enumerate(characters)}
print("char_to_labels:",char_to_labels)

labels_to_char = {val:key for key, val in char_to_labels.items()}
print("labels_to_char:",labels_to_char)
```

Dicha base de datos consta de 1017 muestras de las cuales se dividen 915 para entrenamiento y 102 para su evaluación después de haber sido entrenado con el modelo de redes neuronales convolucionales, como se muestra en la Figura 21.

Figura 21

Resultado del algoritmo

```
Numero de muestras para entrenamiento 915
Numero de muestras para validación 102
char_to_labels: {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6, 'H':
7, 'I': 8, 'J': 9, 'K': 10, 'L': 11, 'M': 12, 'N': 13, 'O': 14, 'P': 15, 'Q': 1
6, 'R': 17, 'S': 18, 'T': 19, 'U': 20, 'V': 21, 'W': 22, 'X': 23, 'Y': 24, 'Z':
25}
labels_to_char: {0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5: 'F', 6: 'G', 7:
'H', 8: 'I', 9: 'J', 10: 'K', 11: 'L', 12: 'M', 13: 'N', 14: 'O', 15: 'P', 16:
'Q', 17: 'R', 18: 'S', 19: 'T', 20: 'U', 21: 'V', 22: 'W', 23: 'X', 24: 'Y', 2
5: 'Z'}
```

En la Figura 22 muestra un script que es una clase de generador de datos personalizada que genera lotes de datos de entrada y las etiquetas correspondientes para entrenar un modelo de red neuronal. Los datos de entrada son un conjunto de imágenes captcha y las etiquetas correspondientes son el texto contenido dentro de esas imágenes.

El generador toma varios argumentos como el conjunto de datos, las etiquetas correspondientes, char_map, el tamaño del lote, las dimensiones de la imagen, la longitud máxima de captcha, el factor de reducción de la muestra, etc. y genera lotes de datos para entrenar el modelo de red neuronal, a continuación, se detalla la descripción del algoritmo.

Algoritmo del Generador de Datos

- Inicialización:
  - Entrada: data, labels, char_map, batch_size (16), img_width (200), img_height (80), downsample_factor (4), max_length (4), shuffle (True)
  - Asigna las variables de entrada a los atributos del objeto.

- Crea una lista de índices con valores desde 0 hasta len(data) - 1.
- Llama a la función on_epoch_end().

- Definición del método __len__:
  - Salida: Número de lotes por época, calculado como el techo de la división de len(data) por batch_size.
- Definición del método __getitem__:
  - Entrada: idx (índice del lote)
  - Calcula curr_batch_idx como los índices del lote actual.
  - Calcula batch_len como la longitud de curr_batch_idx.
  - Inicializa batch_images, batch_labels, input_length, y label_length con las dimensiones y valores adecuados.
  - Para cada índice en curr_batch_idx:
    1. Obtén la imagen correspondiente y transpónla.
    2. Añade una dimensión extra a la imagen.
    3. Obtén la etiqueta correspondiente.
    4. Si la etiqueta es válida:
       - Convierte la etiqueta a una lista de índices usando char_map.
       - Si la longitud de la etiqueta es mayor que max_length, salta esta iteración.
       - Asigna la imagen transpuesta a batch_images.
       - Asigna la etiqueta convertida a batch_labels.

- Asigna la longitud de la etiqueta a label_length.

- Crea un diccionario batch_inputs con las entradas procesadas.
- Devuelve batch_inputs y un array de ceros con longitud batch_len.

- Definición del método on_epoch_end:
  - Si shuffle es verdadero, baraja los índices indices.

Detalle del Algoritmo:

- Inicialización:
  - Al iniciar la clase, se guardan los parámetros de entrada como atributos del objeto.
  - Se crean los índices de los datos y se llama a on_epoch_end() para barajarlos si es necesario.
- Método __len__:
  - Calcula el número total de lotes necesarios para procesar todos los datos con el tamaño de lote dado.
- Método __getitem__:
  - Calcula los índices de los datos para el lote actual.
  - Inicializa las matrices para las imágenes del lote, las etiquetas, la longitud de entrada y la longitud de etiqueta.
  - Para cada índice en el lote:
    - Obtiene y transpone la imagen correspondiente.
    - Añade una dimensión extra para el canal de color.
    - Obtiene la etiqueta correspondiente y verifica si es válida.

        - Convierte la etiqueta a índices y la asigna a las matrices correspondientes si su longitud es menor o igual a max_length.

    - Devuelve las matrices de entrada y un array de ceros.

- Método on_epoch_end:

    - Baraja los índices si se especificó shuffle.

Figura 22

Codificación de la Clase DataGenerator

```
class DataGenerator(keras.utils.Sequence):

    def __init__(self,
                 data,
                 labels,
                 char_map,
                 batch_size=16,
                 img_width=200,
                 img_height=80,
                 downsample_factor=4,
                 max_length=4,
                 shuffle=True
                ):
        self.data = data
        self.labels = labels
        self.char_map = char_map
        self.batch_size = batch_size
        self.img_width = img_width
        self.img_height = img_height
        self.downsample_factor = downsample_factor
        self.max_length = max_length
        self.shuffle = shuffle
        self.indices = np.arange(len(data))
        self.on_epoch_end()

    def __len__(self):
        return int(np.ceil(len(self.data) / self.batch_size))

    def __getitem__(self, idx):
        curr_batch_idx = self.indices[idx*self.batch_size:
(idx+1)*self.batch_size]
        print("curr_batch_idx:",curr_batch_idx)

        batch_len = len(curr_batch_idx)

        batch_images = np.ones((batch_len, self.img_width, self.img_height, 1),
                               dtype=np.float32)
        batch_labels = np.ones((batch_len, self.max_length), dtype=np.float32)
        input_length = np.ones((batch_len, 1), dtype=np.int64) * \
                        (self.img_width // self.downsample_factor - 2)
        label_length = np.zeros((batch_len, 1), dtype=np.int64)


        for j, idx in enumerate(curr_batch_idx):

            print("j;",j,"idx:",idx )
             img = self.data[idx].T
             img = = np.expand_dims(img, axis=-1)

             text = self.labels[idx]
              if is_valid_captcha(text):
                  label = [self.char_map[ch] for ch in text]
                  if len(label)> 4:
                      continue
                  batch_images[j] = img
                  batch_labels[j] = label
                  label_length[j] = len(text)


          batch_inputs = {
                  'input_data': batch_images,
                  'input_label': batch_labels,
                  'input_length': input_length,
                  'label_length': label_length,
                  }
          return batch_inputs, np.zeros(batch_len).astype(np.float32)

        def on_epoch_end(self):
         if self.shuffle:
            np.random.shuffle(self.indices)
```

En la Figura 23 se crea la clase CTCLayer que define una capa personalizada que se utiliza para calcular la función de pérdida CTC (Connectionist Temporal Classification). CTC es una técnica utilizada

para entrenar modelos de reconocimiento de voz y OCR que manejan secuencias de entrada de longitud variable.

La capa CTCLayer se utiliza en la función build_model para calcular la pérdida del modelo. La función de pérdida CTC se utiliza para medir la distancia entre las etiquetas verdaderas y las predicciones del modelo.

Algoritmo de la Capa CTC

- Inicialización:
  - Entrada: name (opcional, nombre de la capa).
  - Llama al constructor de la clase base (layers.Layer) con el nombre dado.
  - Define self.loss_fn como la función de costo CTC (keras.backend.ctc_batch_cost).
- Definición del método call:
  - Entrada: y_true (etiquetas verdaderas), y_pred (predicciones), input_length (longitud de las secuencias de entrada), label_length (longitud de las etiquetas)
  - Calcula la pérdida utilizando self.loss_fn con los parámetros de entrada.
  - Añade la pérdida a la capa usando self.add_loss (loss).
  - Salida: Devuelve la pérdida calculada.

Detalle del Algoritmo:

- Inicialización:

- Al iniciar la capa, llama al constructor de la clase base y guarda la función de costo CTC en self.loss_fn.

- Método call:
  - Paso 1: Calcula la pérdida de CTC utilizando las etiquetas verdaderas, las predicciones, la longitud de las entradas y la longitud de las etiquetas.
  - Paso 2: Añade la pérdida calculada a la capa para que se tenga en cuenta durante el entrenamiento.
  - Paso 3: Devuelve la pérdida calculada para que pueda ser utilizada durante la prueba.

Figura 23

Codificación de la clase CTCLayer

```
class CTCLayer(layers.Layer):
    def __init__(self, name=None):
        super().__init__(name=name)
        self.loss_fn = keras.backend.ctc_batch_cost

    def call(self, y_true, y_pred, input_length, label_length):
        loss = self.loss_fn(y_true, y_pred, input_length, label_length)
        self.add_loss(loss)

        return loss
```

Luego se crea la función build_model define un modelo de OCR utilizando capas convolucionales, capas de reducción de tamaño, capas de transformación de forma, capas densas y capas de LSTM bidireccionales. La salida del modelo se alimenta a la capa personalizada CTCLayer para calcular la pérdida. Finalmente, el modelo se compila y devuelve.

Algoritmo para Construir el Modelo OCR

- Definir Entradas del Modelo:
  - input_img: entrada para las imágenes, con forma (img_width, img_height, 1), nombre 'input_data', tipo float32.
  - labels: entrada para las etiquetas, con forma [max_length], nombre 'input_label', tipo float32.
  - input_length: entrada para la longitud de las secuencias de entrada, con forma [1], nombre 'input_length', tipo int64.
  - label_length: entrada para la longitud de las etiquetas, con forma [1], nombre 'label_length', tipo int64.
- Primer bloque de convolución:
  - Aplica una capa de convolución 2D con 32 filtros, un kernel de tamaño (3, 3), activación 'relu', inicializador 'he_normal', y padding 'same'. Nombre de la capa: 'Conv1'.
  - Aplica una capa de max pooling con tamaño (2, 2). Nombre de la capa: 'pool1'.
- Segundo bloque de convolución:
  - Aplica una capa de convolución 2D con 64 filtros, un kernel de tamaño (3, 3), activación 'relu', inicializador 'he_normal', y padding 'same'. Nombre de la capa: 'Conv2'.
  - Aplica una capa de max pooling con tamaño (2, 2). Nombre de la capa: 'pool2'.
- Reducción de Dimensionalidad:

  - Calcula la nueva forma de los mapas de características después de dos capas de max pooling (cada una con tamaño y stride de 2), resultando en mapas de características 4 veces más pequeños en cada dimensión espacial.
  - La nueva forma es ((img_width // 4), (img_height // 4) * 64).

- Reestructuración y Densas:
  - Reestructura los mapas de características a la nueva forma calculada.
  - Aplica una capa densa con 64 unidades y activación 'relu'. Nombre de la capa: 'dense1'.
  - Aplica una capa de dropout con una tasa de 0.2.
- Capas RNN:
  - Aplica una capa bidireccional de LSTM con 128 unidades, retorno de secuencias y dropout de 0.2.
  - Aplica una segunda capa bidireccional de LSTM con 64 unidades, retorno de secuencias y dropout de 0.25.
- Cálculo de Predicciones:
  - Aplica una capa densa con un número de unidades igual a len(characters) + 1 y activación 'softmax'. Nombre de la capa: 'dense2'.
- Cálculo de Pérdida CTC:
  - Usa la capa personalizada CTCLayer para calcular la pérdida CTC con las entradas: labels, x (predicciones), input_length y label_length.

- Definición del Modelo:
  - Define el modelo con las entradas [input_img, labels, input_length, label_length] y la salida output.
  - Asigna un nombre al modelo: 'ocr_model_v1'.
- Optimizador:
  - Define el optimizador SGD con una tasa de aprendizaje de 0.002, decaimiento 1e-6, momentum 0.9, Nesterov activado y clipnorm de 5.
- Compilación del Modelo:
  - Compila el modelo usando el optimizador SGD y la métrica 'accuracy'.
- Salida:
  - Devuelve el modelo compilado.

Figura 24

Función Build_model

```
def build_model():
    input_img = layers.Input(shape=(img_width, img_height, 1),
                             name='input_data',
                             dtype='float32')
    labels = layers.Input(name='input_label', shape=[max_length], dtype='float32')
    input_length = layers.Input(name='input_length', shape=[1], dtype='int64')
    label_length = layers.Input(name='label_length', shape=[1], dtype='int64')

    x = layers.Conv2D(32,
                (3,3),
                activation='relu',
                kernel_initializer='he_normal',
                padding='same',
                name='Conv1')(input_img)
    x = layers.MaxPooling2D((2,2), name='pool1')(x)

    x = layers.Conv2D(64,
                (3,3),
                activation='relu',
                kernel_initializer='he_normal',
                padding='same',
                name='Conv2')(x)
    x = layers.MaxPooling2D((2,2), name='pool2')(x)

    new_shape = ((img_width // 4), (img_height // 4)*64)
    print("new_shape:",new_shape)

    x = layers.Reshape(target_shape=new_shape, name='reshape')(x)
    x = layers.Dense(64, activation='relu', name='dense1')(x)
    x = layers.Dropout(0.2)(x)

    x = layers.Bidirectional(layers.LSTM(128,
                                         return_sequences=True,
                                         dropout=0.2))(x)
    x = layers.Bidirectional(layers.LSTM(64,
                                         return_sequences=True,
                                         dropout=0.25))(x)

    x = layers.Dense(len(characters)+1,
         activation='softmax', name='dense2', kernel_initializer='he_normal')(x)

    output = CTCLayer(name='ctc_loss')(labels, x, input_length, label_length)

    model = keras.models.Model(inputs=[input_img,labels,
input_length,label_length],     outputs=output, name='ocr_model_v1')

    sgd = keras.optimizers.SGD(learning_rate=0.002,
                               decay=1e-6,
                               momentum=0.9,
                               nesterov=True,
                               clipnorm=5)

    model.compile(optimizer=sgd,metrics=['accuracy'])
    return model
```

La arquitectura propuesta que se presenta en la Figura 24, es gracias a la creación de una instancia de un modelo de red neuronal convolucional definido por la función model = build_model(), luego la función model.summary() muestra toda la información de dicha arquitectura de la instancia, como también el número total de parámetros del modelo y el tamaño de su salida como se muestra en la Tabla 7.

Tabla 6

Arquitectura de la red neuronal convolucional

Layer (Type)	Output Shape	Param	Connected to
Input_Data (Inputlayer)	[(None, 200, 80, 1)]	0	[]
Conv1 (Conv2d)	(None, 200, 80, 32)	320	['input_data[0][0]']
Pool1 (Maxpooling2d)	(None, 100, 40, 32)	0	['Conv1[0][0]']
Conv2 (Conv2d)	(None, 100, 40, 64)	18496	['pool1[0][0]']
Pool2 (Maxpooling2d)	(None, 50, 20, 64)	0	['Conv2[0][0]']
Reshape (Reshape)	(None, 50, 1280)	0	['pool2[0][0]']
Dense1 (Dense)	(None, 50, 64)	81984	['reshape[0][0]']
Dropout (Dropout)	(None, 50, 64)	0	['dense1[0][0]']
Bidirectional (Bidirectional)	(None, 50, 256)	197632	['dropout[0][0]']
Bidirectional_1 (Bidirectional)	(None, 50, 128)	164352	['bidirectional[0][0]']
Input_Label (Inputlayer)	[(None, 4)]	0	[]

Layer (Type)	Output Shape	Param	Connected to
Dense2 (Dense)	(None, 50, 27)	3483	['bidirectional_1[0][0]']
Input_Length (Inputlayer)	[(None, 1)]	0	[]
Label_Length (Inputlayer)	[(None, 1)]	0	[]
Ctc_Loss (Ctclayer)	(None, 1)	0	['input_label[0][0]',
			'dense2[0][0]',
			'input_length[0][0]',
			'label_length[0][0]']
Total Params: 466,267			
Trainable Params: 466,267			
Non-Trainable Params: 0			

En la Figura 25 se muestra una codificación que entrena el modelo creado con los generadores de datos ‘train_data_generator’ y ‘valid_data_generator’. Utiliza la función fit de Keras para entrenar el modelo con un número de épocas igual a 50 y utiliza es como un callback de early stopping. es detendrá el entrenamiento si la función de pérdida en el conjunto de validación no mejora después de 5 épocas consecutivas.

Figura 25

Función Kit de Keras

```
es = keras.callbacks.EarlyStopping(monitor='val_loss',
                                   patience=5,

restore_best_weights=True)
history = model.fit(train_data_generator,
                    validation_data=valid_data_generator,
                    epochs=50,
                    callbacks=[es])
```

4.1.3. Evaluar la exactitud del captcha mediante una métrica de error en un entorno controlado

Luego de haber construido la arquitectura del modelo se procede a realizar el entrenamiento en esta etapa se pretende minimizar la función error que se muestra en la Figura 26 y maximizar la precisión o exactitud del modelo que se denota en la Figura 27, estos se pueden visualizar en lo para lo cual se puede visualizar el comportamiento del entrenamiento en la Figura 28.

Figura 26

Función mínima del error

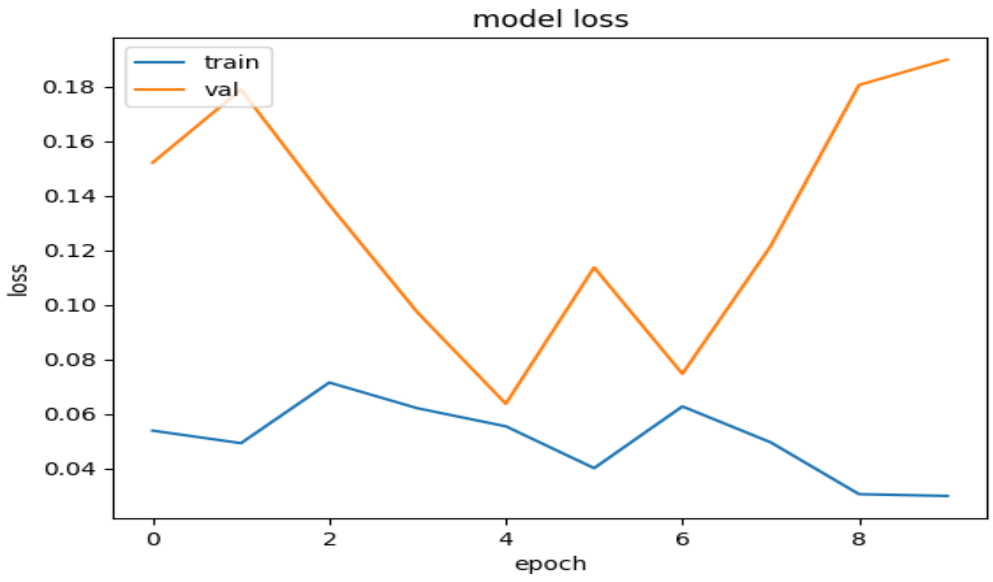
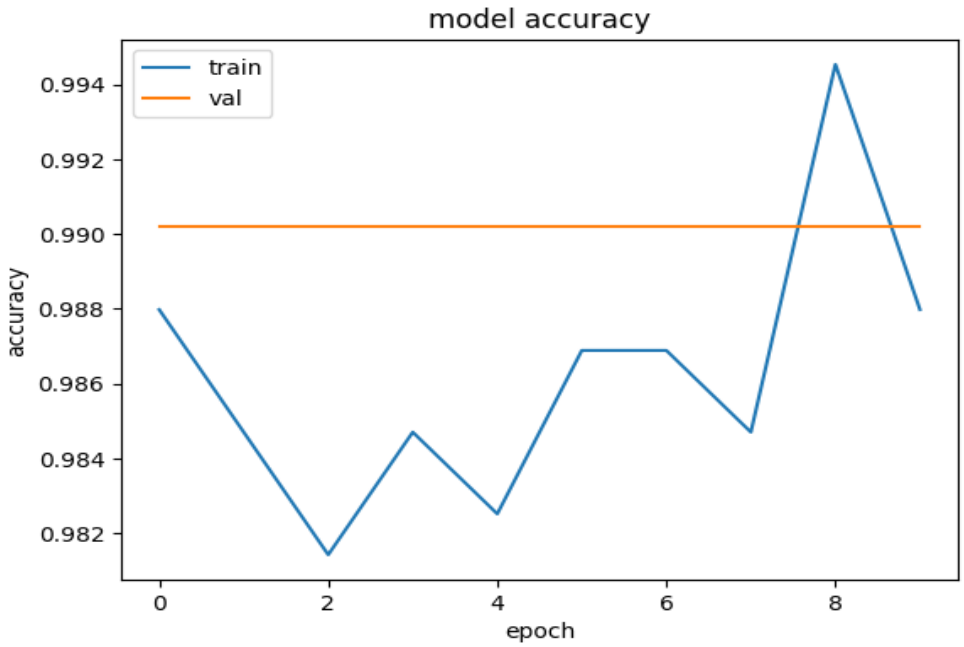


Figura 27

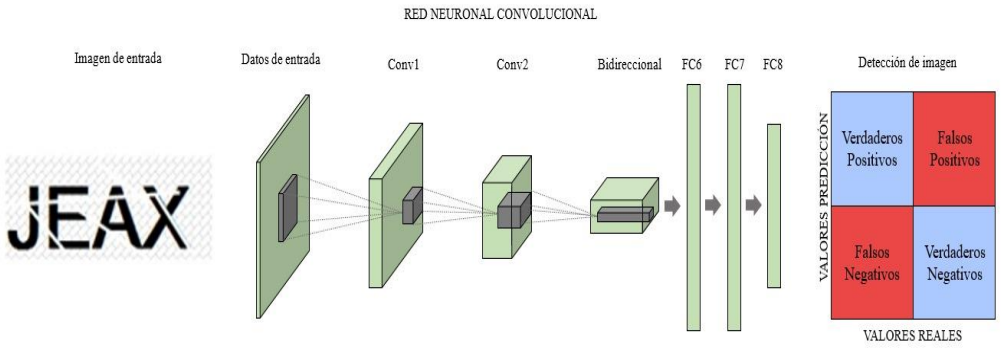
Función máxima de la precisión



Entendiendo que se está abordando un problema de reconocimiento óptico de caracteres se ha visto por conveniente utilizar las de exactitud que nos da una idea global de comportamiento de este modelo de inteligencia artificial, teniendo como resultado este resultado significa que el 97.05% de veces tendrá un resultado correcto.

Figura 28

Modelo para el reconocimiento de CAPTCHA's



En la Figura 29 se están importando las funciones ‘accuracy_score’ y ‘precisión_score’ del módulo ‘sklearn.metrics’, donde ‘y_pred’ (predicciones) se asigna a la variable ‘pred_texts’ y ‘y_true’ (valores verdaderos) se asigna a la variable ‘orig_texts’. Esto supone que ‘pred_texts’ y ‘orig_texts’ son listas o arrays que contienen las predicciones del modelo y los valores verdaderos, respectivamente. La función ‘accuracy_score’ calcula la exactitud de las predicciones comparando ‘y_true’ con ‘y_pred’. La exactitud es la proporción de predicciones correctas sobre el total de predicciones la función ‘precision_score’ calcula la precisión de las predicciones. La precisión mide la proporción de verdaderos positivos entre los positivos predichos (es decir, qué tan precisa es la clasificación del modelo en términos de identificar correctamente las instancias positivas

Figura 29

Script de cálculo de accuracy y precision

```
In [24]: from sklearn.metrics import accuracy_score
         from sklearn.metrics import precision_score

         y_pred = pred_texts
         y_true = orig_texts

         acc = accuracy_score(y_true, y_pred)
         prec = precision_score(y_true, y_pred, average="macro")

C:\Users\ADMIN\.conda\envs\tensorflow\lib\site-packages\sklearn\metrics\_classification.py:1327: UndefinedMetricWarning: Precis
ion is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
```

En la cual dado el anterior script vamos a imprimir las variables “acc” y “prec” donde muestre los valores calculados en la Tabla 8 de un proceso estocástico donde dichos valores calculados pueden variar en cada entrenamiento dado que se muestra en la Figura 30.

Figura 30

Impresión de resultados

Tabla 7

Resultados de exactitud y precisión

Accuracy:	0.9901960784313726
Prec:	0.9805825242718447

4.2. DISCUSIÓN

La investigación realizada por Walia & odugoudar (2023) y Villares-Jimenez & Quinatoa-Medina (2023) destaca la importancia de abordar la vulnerabilidad de los sistemas CAPTCHA desde diferentes perspectivas. Mientras que Walia & odugoudar se centran en el análisis de la vulnerabilidad utilizando técnicas de aprendizaje profundo, específicamente redes neuronales convolucionales (CNN), Villares-Jimenez Julio & Quinatoa-Medina exploran la elusión de los sistemas CAPTCHA mediante la combinación de técnicas de visión artificial y procesos de bot automatizados. Estos estudios resaltan la evolución de las amenazas cibernéticas y la necesidad de estrategias innovadoras para enfrentarlas.

Por otro lado, los resultados de la investigación presentada ofrecen una contribución significativa al campo al presentar un modelo efectivo para el reconocimiento de CAPTCHA basado en redes neuronales convolucionales. La precisión

obtenida por el modelo desarrollado, del 97.05%, es notablemente superior a la precisión del 35% alcanzada por el modelo basado en procesamiento de imágenes. Esto subraya la eficacia de las redes neuronales convolucionales en la tarea de reconocimiento de CAPTCHA, respaldando la literatura existente que indica mejores resultados con este enfoque.

Sin embargo, es importante reconocer que, a pesar de la disponibilidad de tecnologías como la desarrollada en la presente investigación, la seguridad en las entidades públicas y privadas peruanas sigue siendo un aspecto poco considerado. Existe una brecha significativa entre la disponibilidad de soluciones tecnológicas y su implementación efectiva en entornos gubernamentales, lo que deja a estas instituciones vulnerables en supuestos ataques cibernéticos. Además, el aumento constante de la tecnología y la inteligencia artificial sugiere que habrá un incremento en la sofisticación y frecuencia de los ataques cibernéticos en el futuro.

V. CONCLUSIONES

Se presenta un modelo para reconocimiento de CAPTCHA, para lo cual se utiliza una red neuronal convolucional, ya que la literatura indica que tiene mejores resultados en imágenes.

- - Se obtuvo una base de datos de captchas con 1017 imágenes las cuales se utilizaron para el entrenamiento del modelo, en el cual se utilizó dado el contexto de este desarrollo de un modelo de inteligencia artificial, los métodos y las técnicas que incluyen la preparación y manipulación de datos, la selección y división de datos en conjuntos de entrenamiento y su validación, así como también el uso de bibliotecas específicas como OpenCV para el preprocesamiento de datos y se puso en marcha todo el proceso con la creación de una tabla de datos.
- Se implementó y se entrenó con éxito los dos modelos para el reconocimiento óptico de caracteres y otro basado en procesamiento de imágenes. Se pudo determinar que el modelo basado en redes neuronales convolucionales converge de mejor manera durante el entrenamiento con los métodos, técnicas de aprendizaje automático y aprendizaje profundo con la importación de librerías, todo ello codificado en el lenguaje Python dentro de un entorno controlado como Virtualenv y/o un IDE en este caso el software Anaconda Navigator.
- Se evaluó con éxito las dos propuestas, un primer modelo basado en procesamiento de imágenes, con el que se obtuvo una exactitud de 35%, pero con el otro modelo basado en redes neuronales convolucionales se obtuvo una exactitud de 97.05%, mostrando mejores resultados, con todo ello se utilizó en la implementación de estos dos modelos utilizando una matriz de confusión, el método F1-score, exactitud, precisión, sensibilidad y especificidad.

VI. RECOMENDACIONES

- Se recomienda que la base de datos sea diverso y representativo de los diferentes tipos de captchas que el modelo pueda ser útil en una aplicación web, por lo que se recomienda hacer una actualización de captchas en dichas aplicaciones o sistemas de información lo que ayuda a garantizar que el modelo sea robusto frente a diversas variaciones y estilos.

- Por otro lado se recomienda experimentar con diferentes configuraciones de hiper parámetros, como tasas de aprendizaje, arquitecturas de red y funciones de pérdida como LTSM. Ajustar estos parámetros puede tener un impacto significativo en el rendimiento del modelo.

- Se recomienda evaluar captchas de las diversas páginas de las instituciones que conforma el Estado Peruano, así como otras páginas y ver que tan vulnerable es su sistema de verificación de captchas.

VII. REFERENCIAS BIBLIOGRÁFICAS

Aguilar Fauta, D. A., & others. (2021). *Text-based CAPTCHA Vulnerability Assessment using Deep Learning-based Solver*. Quito.

Bekkar, M., Djema, H., & Alitouche, T. A. (2013). *Evaluation measures for models assessment over imbalanced data sets. Journal of Information Engineering and Applications, 3, 27–38.*

Bosch, A., Roma, J. C., & Bagén, A. L. (2019). *Deep learning: principios y fundamentos*. Editorial UOC.

Calizaya Bobadilla M. R., & Calsin Cari. F. (2023). *Modelo para la detección de anomalías en secuencias de videos de exámenes en línea mediante inteligencia artificial caso de estudio: Universidad Nacional del Altiplano*. https://repositorio.unap.edu.pe/handle/20.500.14082/20793

Cheung, S. S., & Kamath, C. (2004). *Robust techniques for background subtraction in urban traffic video (S. Panchanathan & B. Vasudev*, Eds.; p. 881). https://doi.org/10.1117/12.526886

Chollet, F. (2017). *Deep learning with Python.* New York: Manning Publications.

Cortés, J. A., Muriel, A., & Mendoza, J. A. (2011). *Qualitative and quantitative Comparison of Basic histogram-based global Thresholding techniques for Digital Image Processing*. Scientia et Technica, 3(49), 266–272.

Cuevas, E., Cortés, M. D., & Méndez, J. O. C. (2018). *Tratamiento de imágenes con MATLAB*. Marcombo.

Dankwa, S., & Yang, L. (2021). A*n efficient and accurate depth-wise separable convolutional neural network for cybersecurity vulnerability assessment based on CAPTCHA breaking*. Electronics, 10(4), 480.

Davis, J., & Goadrich, M. (2006). *The relationship between Precision-Recall and ROC curves.* Proceedings of the 23rd International Conference on Machine Learning, 233–240.

Dawson, M. (2010). *Python programming for the absolute beginner*. Course Technology Boston, MA.

Escobar Tafurt, L. F. (2016). *Evaluación de algoritmos de sustracción de fondo para conteo de personas*. Universidad del Valle. https://hdl.handle.net/10893/15687

Garcia Serrano, E. D. (n.d.). *Modelos de captcha basados en habilidades de agentes artificiales*. Retrieved January 6, 2024, from http://tesis.ipn.mx/handle/123456789/25390

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

Hasan, W. K. (2016). *A survey of current research on captcha.* International Journal of Computer Science & Engineering Survey, 7(3), 141-157.

Heras, J. M. (2020). *Precision, recall, F1, accuracy en clasificación*. IArtificial. Net.

Hu, Y., Chen, L., & Cheng, J. (2018). *A CAPTCHA recognition technology based on deep learning*. 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), 617–620. https://doi.org/10.1109/ICIEA.2018.8397789

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

Lin, D., Lin, F., Lv, Y., Cai, F., & Cao, D. (2018). *Chinese character CAPTCHA recognition and performance estimation via deep neural network.* Neurocomputing, 288, 11–19.

Ling-Zi, X., & Yi-Chun, Z. (2012). *A Case Study of Text-Based CAPTCHA Attacks.* 2012 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 121–124. https://doi.org/10.1109/CyberC.2012.28

Malpartida, E. A. S. (2011). *Sistema de visión artificial para el reconocimiento y manipulación de objetos utilizando un brazo robot*. Pontificia Universidad Católica del Perú (Perú).

McKinney, W. (2022). *Python for data analysis*. " O'Reilly Media, Inc."

Méndez, J. T. P., Morales, R. M., & others. (2008). *Inteligencia Artificial: Métodos, técnicas y aplicaciones*. McGraw-Hill Interamericana.

Mendez, K. M., Pritchard, L., Reinke, S. N., & Broadhurst, D. I. (2019). *Toward collaborative open data science in metabolomics using Jupyter Notebooks and cloud computing.* Metabolomics, 15, 1–16.

Minguez, T. D. (2021). *Visión artificial: aplicaciones prácticas con OpenCV-Python*. Marcombo.

Mokhtari, S., Yen, K. K., & Liu, J. (2021). *Effectiveness of Artificial Intelligence in Stock Market Prediction based on Machine Learning*. International Journal of Computer Applications, 183(7), 1–8. https://doi.org/10.5120/ijca2021921347

Moralejo, R. O., & Storni, A. (2018). *Plataforma Detección de objetos en tiempo real.* XX Workshop de Investigadores en Ciencias de la Computación (págs. 371-375). Universidad Nacional del Nordeste - Argentina: Red de Universidades con Carreras de Informática.

Mousalli-Kayat, G. (2015). *Métodos y diseños de investigación cuantitativa*. Mérida.

Noury, Z., & Rezaei, M. (2020). *Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment*. ArXiv Preprint ArXiv:2006.08296.

Ñaupas Paitán, H. V. D. M. R. P. V. J. J. R. D. H. E. (2018). *Metodología de la Investigación cuantitativa - cualitativa y Redacción de la Tesis*.

Powers, D. M. W. (2020). *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.* ArXiv Preprint ArXiv:2010.16061.

Puig, I., López-Cerón, M., Arnau, A., Rosiñol, Ò., Cuatrecasas, M., Herreros-de-Tejada, A., Ferrández, Á., Serra-Burriel, M., Nogales, Ó., Vida, F., de Castro, L., López-Vicente, J., Vega, P., Álvarez-González, M. A., González-Santiago,

J., Hernández-Conde, M., Díez-Redondo, P., Rivero-Sánchez, L., Gimeno-García, A. Z., … Ascon, N. (2019). *Accuracy of the Narrow-Band Imaging International Colorectal Endoscopic Classification System in Identification of Deep Invasion in Colorectal Polyps.* Gastroenterology, 156(1), 75–87. https://doi.org/10.1053/j.gastro.2018.10.004

Reitz, K., & Schlusser, T. (2016). *The Hitchhiker's guide to Python: best practices for development*. " O'Reilly Media, Inc."

Rivera Demanuel, D. R., Huamani Huancara, C., & Charca Ccama, Y. A. (2022). *Sistema automático para calificación de vino mediante Redes Neuronales.* Innovación y Software, 3(1), 30–46. https://doi.org/10.48168/innosoft.s8.a51

Rolon-Mérette, D., Ross, M., Rolon-Mérette, T., & Church, K. (2016). *Introduction to Anaconda and Python: Installation and setup*. Quant. Methods Psychol, 16(5), S3–S11.

Roncagliolo, P. (2007). *Procesamiento digital de imágenes*.

Russell, S. J., & Norvig, P. (2010). *Artificial intelligence a modern approach*. London.

Sánchez Carlessi, H., Reyes Romero, C., & Mejía Sáenz, K. (2018). *Manual de términos en investigación científica, tecnológica y humanística*.

Sandoval Serrano, L. J. (2018). *Algoritmos de aprendizaje automático para análisis y predicción de datos*. Revista Tecnológica, 11

Sergio Delgado Quintero. (2022). *Aprende Python*.

Smith, R. (2007). *An Overview of the Tesseract OCR Engine.* Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), 2, 629–633. https://doi.org/10.1109/ICDAR.2007.4376991

Somoza Barreiro, M., & Jiménez Gutiérrez, L. M. (2016). *Reconocimiento óptico de caracteres mediante imágenes en contadores de gas*. http://hdl.handle.net/20.500.14352/66041

Stark, F., Hazrbas, C., Triebel, R., & Cremers, D. (2015). *Captcha recognition with active deep learning*. Workshop New Challenges in Neural Computation, 2015, 94.

Szeliski, R. (2011). *Computer Vision*. Springer London. https://doi.org/10.1007/978-1-84882-935-0

Taquía-Gutiérrez, J. A. (2017). *El procesamiento de imágenes y su potencial aplicación en empresas con estrategia digital*. Interfases, 0(010), 11. https://doi.org/10.26439/interfases2017.n10.1767

Tharwat, A. (2021). *Classification assessment methods*. Applied Computing and Informatics, 17(1), 168–192. https://doi.org/10.1016/j.aci.2018.08.003

Torres, J. (2020). *Python deep learning: Introducción práctica con Keras y TensorFlow 2*. Alpha Editorial.

Turing, A. M. (2009). *Computing machinery and intelligence*. Springer.

Van Rijsbergen, C. (1979). *Information retrieval: theory and practice. Proceedings of the Joint IBM/University of Newcastle upon Tyne Seminar on Data Base Systems*, 79.

Villares-Jimenez Julio and Quinatoa-Medina, J. and R.-G. G. and N.-A. D. and S.-T. B. (2023). *Vulnerability of CAPTCHA Systems Using Bots with Computer Vision Abilities.* In M. and M. L. S. and T.-C. P. and D. B. Botto-Tobar Miguel and Zambrano Vizuete (Ed.), Applied Technologies (pp. 329–343). Springer Nature Switzerland.

Walia, J. S., & odugoudar, A. (2023). *Vulnerability analysis of captcha using Deep learning*.

Wang, J., Qin, J., Xiang, X., Tan, Y., & Pan, N. (2019). *CAPTCHA recognition based on deep convolutional neural network*. Mathematical Biosciences and Engineering, 16(5), 5851–5861. https://doi.org/10.3934/mbe.2019292

Wang, Y., & Lu, M. (2018). *An optimized system to solve text-based CAPTCHA*.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data mining: Practical machine learning tools and techniques* (3rd ed.). Morgan Kaufmann.

Wu, Y. (2020). *Is a dataframe just a table?* OpenAccess Series in Informatics, 76. https://doi.org/10.4230/OASIcs.PLATEAU.2019.6

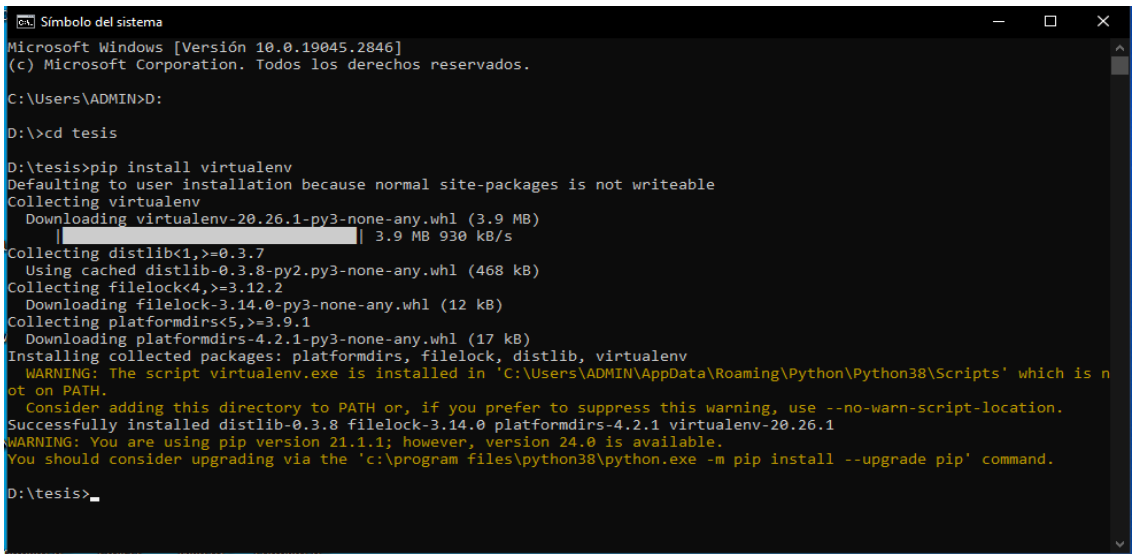
Yamaguchi, M., Nakata, T., Watanabe, H., Okamoto, T., & Kikuchi, H. (2014). *Vulnerability of the conventional accessible CAPTCHA used by the White House and an alternative approach for visually impaired people.* 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 3946–3951. https://doi.org/10.1109/SMC.2014.6974548

Zhou, S., Chen, Q., & Wang, X. (2013). *Active deep learning method for semi-supervised sentiment classification*. https://doi.org/10.1016/j.neucom.2013.04.017

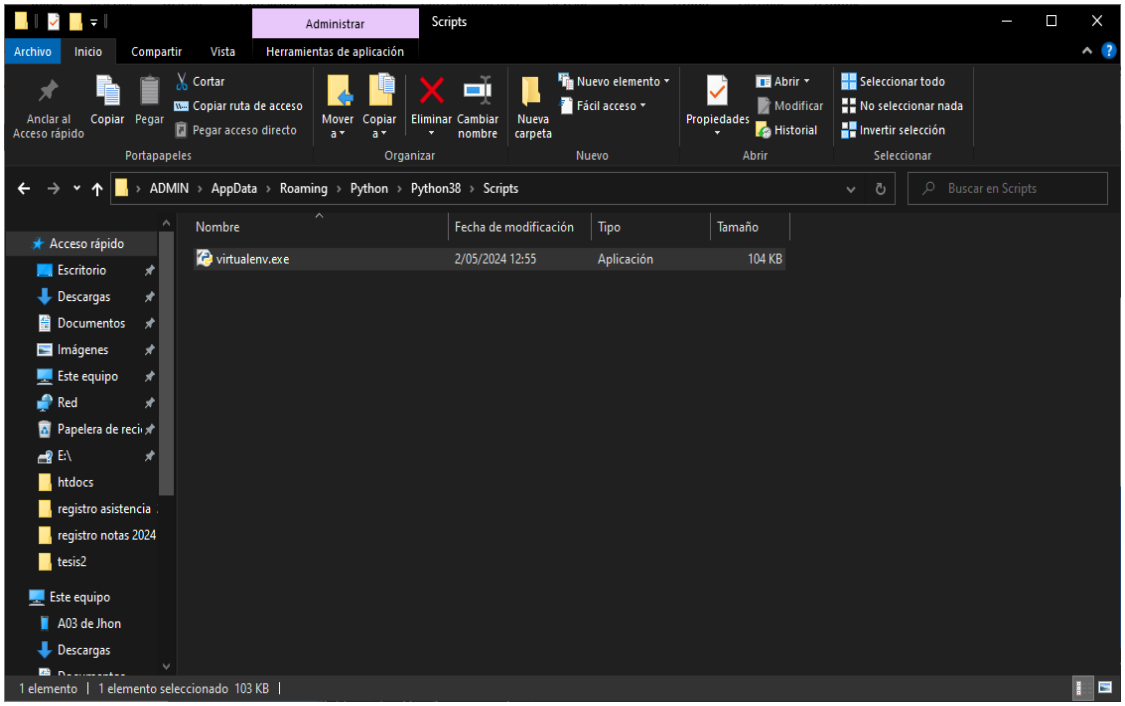
Zhu, B. B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., & Cai, K. (2010). *Attacks and design of image recognition CAPTCHAs*. Proceedings of the 17th ACM Conference on Computer and Communications Security, 187–200. https://doi.org/10.1145/1866307.1866329

ANEXOS

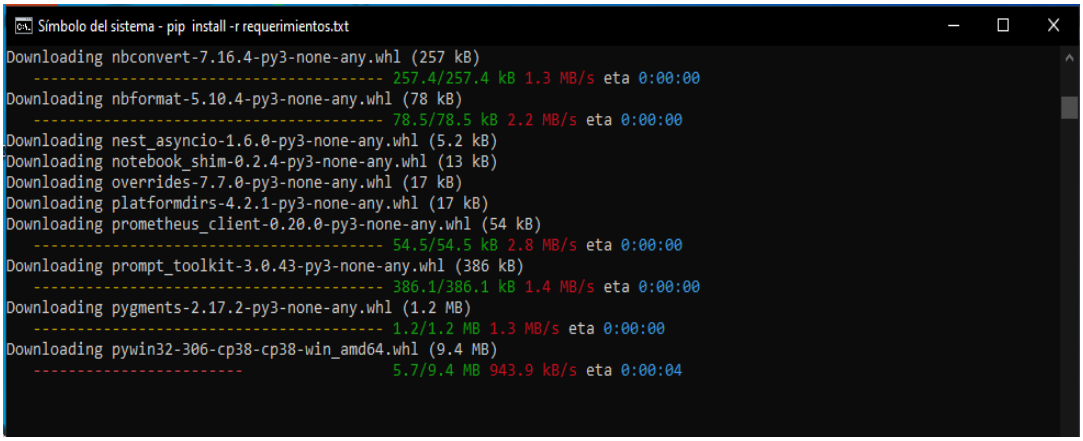
ANEXO 1: Instalación de Virtualenv



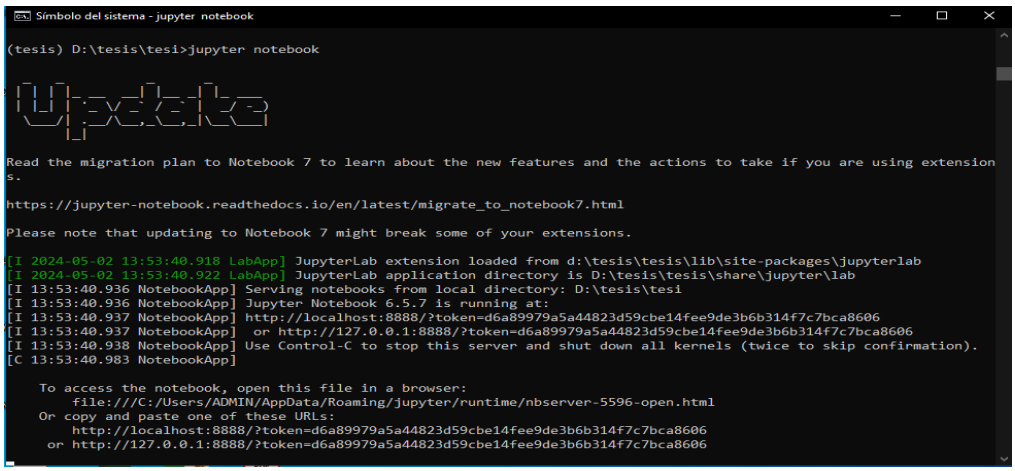
ANEXO 2: Carpeta del paquete Virtualenv



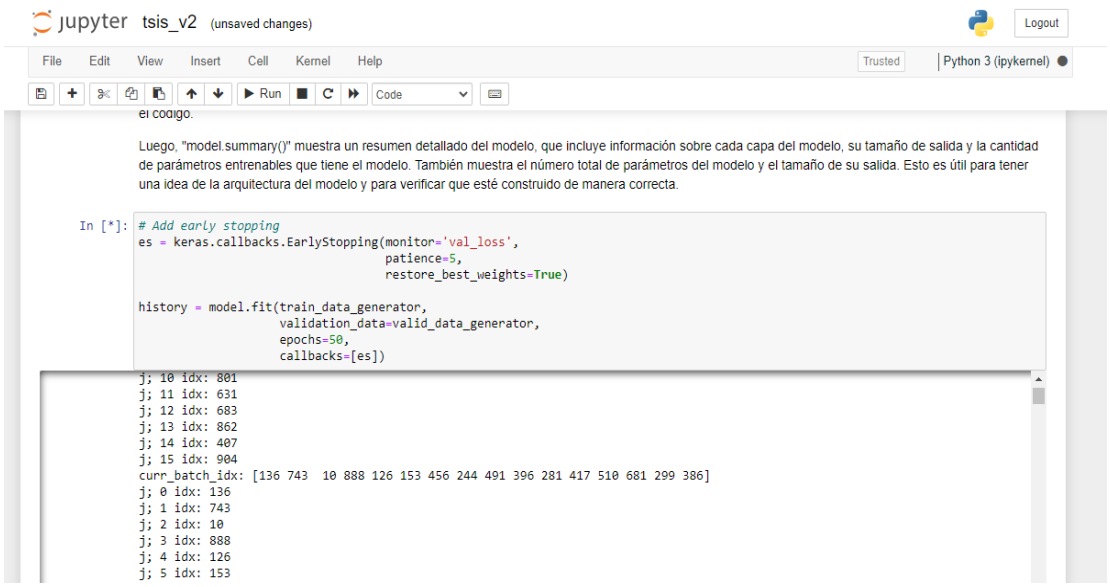
ANEXO 3: Instalación de requerimientos de paquetes Python



ANEXO 4: Ejecución del programa Jupyter Notebook.



ANEXO 5: Compilación de scripts.



ANEXO 6: Declaración jurada de autenticidad de tesis.

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

DECLARACIÓN JURADA DE AUTENTICIDAD DE TESIS

Por el presente documento, Yo Jhon Carlos Gonzales Quilca y Elmer Ander Chalco Huarachi, identificado con DNI 70056211 y 76974584 en mi condición de egresado de:

☒ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☐ **Programa de Maestría o Doctorado**
Ingeniería de Sistemas,

informo que he elaborado el/la ☒ **Tesis** o ☐ **Trabajo de Investigación** denominada:
" Evaluación de la vulnerabilidad del captcha mediante el reconocimiento con redes neuronales convolucionales ",

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 15 de octubre del 2024

Jhon Carlos Gonzales Quilca
DNI N° 70056211

Elmer Ander Chalco Huarachi
DNI N° 76974584

ANEXO 7: Autorización para el depósito de tesis en el Repositorio Institucional

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 Jhon Carlos Gonzales Quilca y Elmer Ander Chalco Huarachi, identificado con DNI 70056211 y 76974584 en mi condición de egresado de:

☒ **Escuela Profesional,** ☐ **Programa de Segunda Especialidad,** ☐ **Programa de Maestría o Doctorado**

Ingeniería de Sistemas,

informo que he elaborado el/la ☒ **Tesis** o ☐ **Trabajo de Investigación** denominada:

" Evaluación de vulnerabilidad del captcha mediante el reconocimiento con redes neuronales convolucionales "

para la obtención de ☐**Grado,** ☒ **Título Profesional** o ☐ **Segunda Especialidad.**

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 15 de Octubre del 20 24

Jhon Carlos Gonzales Quilca
DNI N° 70056211

Elmer Ander Chalco Huarachi
DNI N° 76974584