UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

IDENTIFICACIÓN AUTOMÁTICA DE GATOS MEDIANTE RECONOCIMIENTO DE IMÁGENES USANDO REDES NEURONALES CONVOLUCIONALES

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS

Autor:

Bach. Hernandez Neria Marco Antonio Rosas

ORCID: https://orcid.org/0000-0002-4270-7499

Asesor:

Mg. Mejia Cabrera Heber Ivan

ORCID: https://orcid.org/0000-0002-0007-0928

Línea de Investigación:

Infraestructura, Tecnología y Medio Ambiente

Pimentel – Perú 2022

APROBACIÓN DEL JURADO

IDENTIFICACIÓN AUTOMÁTICA DE GATOS MEDIANTE RECONOCIMIENTO DE IMÁGENES USANDO REDES NEURONALES CONVOLUCIONALES

Bach. Hernandez Neria Marco Antonio Rosas

Autor

Mg. Mejía Cabrera Heber Iván

Asesor

Mg. Mejía Cabrera Heber Iván	**Mg. Aguinaga Tello Juan**
Presidente de Jurado	**Secretario de Jurado**

Mg. Minguillo Rubio Cesar Augusto

Vocal de Jurado

Dedicatoria

Dedicado a mis papás Rosas Hernández y Madeleine Neria, por darme la oportunidad de estudiar una carrera, por estar ahí siempre conmigo en los buenos y malos momentos y por siempre darme ese empujón extra que en algunas ocasiones era necesario para continuar y no tirar la toalla. A mi hermana Rosa Hernández que siempre me alentó a seguir adelante y no rendirme, que me enseñó prácticamente a vivir solo y valerme por mí mismo cuando empecé esta etapa universitaria. Y a todos mis seres queridos que de cierta forma me brindaron su apoyo.

Agradecimiento

A Dios por su bendecirme siempre y darme las fuerzas para poder seguir adelante ante las adversidades; a mis papás que siempre creyeron en mí y estuvieron siempre conmigo; a mi asesor de tesis, Mg. Heber Iván Mejía Cabrera, por la motivación, enseñanzas y apoyo que me brindó a lo largo de la carrera.

Resumen

Existe una alta tasa de mascotas domesticas abandonadas en el Perú, las entidades gubernamentales y no gubernamentales luchan cada día para que estos números se reduzcan con diferentes campañas, incentivando a la adopción en hogares que no cuenten con mascotas y también que realicen la esterilización en sus mascotas, pero aparte de estas campañas como combatir el abandono de las mascotas que realizan las personas, y así hacer valer la ley de protección y bienestar animal. Existen métodos para identificar si una mascota ha sido abandonada por ejemplo que sea identificada por el ojo humano, el uso de microchips, el uso de collar con código QR, pero como identificar si estos tres métodos mencionados no funcionan, frente a esta situación, como identificamos a una mascota abandonada en la calle, se trabajó con deep learning utilizando redes neuronales convolucionales, para identificar de manera automática a los gatos desaparecidos. Los resultados que se obtuvieron fueron de un 98% de exactitud al momento de probar la red neuronal convolucional, llegando así a la conclusión que las redes neuronales convolucionales son muy efectivas y fáciles de implementar para realizar trabajos de clasificación y predicción.

Palabras clave: redes neuronales convolucionales, cnn, aprendizaje profundo, reconocimiento de imágenes de gatos.

Abstrac

There is a high rate of domestic pets abandoned in Peru, governmental and non-governmental entities struggle every day to reduce these numbers with different campaigns, encouraging adoption in homes that do not have pets and also perform sterilization in their pets, but apart from these campaigns such as combating the abandonment of pets carried out by people, and thus enforce the law of protection and animal welfare. There are methods to identify if a pet has been abandoned for example that is identified by the human eye, the use of microchips, the use of a collar with QR code, but how to identify if these three methods mentioned do not work, in the face of this situation, such as We identified an abandoned pet in the street, worked with deep learning using convolutional neural networks, to automatically identify missing cats. The results obtained were 98% accuracy at the time of testing the convolutional neural network, thus concluding that the convolutional neural networks are very effective and easy to implement to perform classification and prediction work.

Keywords: convolutionary neural networks, cnn, deep learning, image recognition of cats.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I. INTRODUCCIÓN

1.1 Realidad Problemática

El abandono de mascotas domésticas es un problema que se viene luchando desde siempre con la creación de refugios animales e inclusive en algunos países las personas que abandonan a sus mascotas son severamente castigadas por la ley, ya que estas mascotas cuentan con respaldo legal. En Perú la ley N° 30407 “Ley de protección y bienestar animal” indica que las personas que tengan a cargo a un animal de compañía deben asegurarse de su bienestar y también de su seguridad, brindándole un ambiente adecuado que les permita expresar comportamiento natural, alimentación suficiente y adecuada, atención medica-veterinaria especializada y vacunación de ser necesario.

Día Internacional del Animal Sin hogar: ¿Qué puedes hacer tú para ayudarlos?. (17 de agosto de 2019). *Perú21*. Recuperado de https://peru21.pe/. En una noticia publicada en el diario Perú21 indicó que en El Perú hay 6 millones de perros abandonados y tan solo en Lima hay 2 millones, en cuanto a gatos esta cifra se triplica debido a que su reproducción ocurre con mayor rapidez.

La fundación affinity (2018) en su estudio realizado sobre el desamparo, la pérdida y la adopción de mascotas en España, se llegó a la conclusión que en promedio diecisiete de cada mil perros y diez de cada mil gatos han sido ingresados a refugios en España. En las dos figuras se logra visualizar el aumento de ingreso de estas mascotas a los refugios animales de España a lo largo de los años desde 1998 hasta el 2018. En 2008 fue el año donde se tuvo el mayor aumento de ingresos de mascotas entre gatos y perros a los refugios.

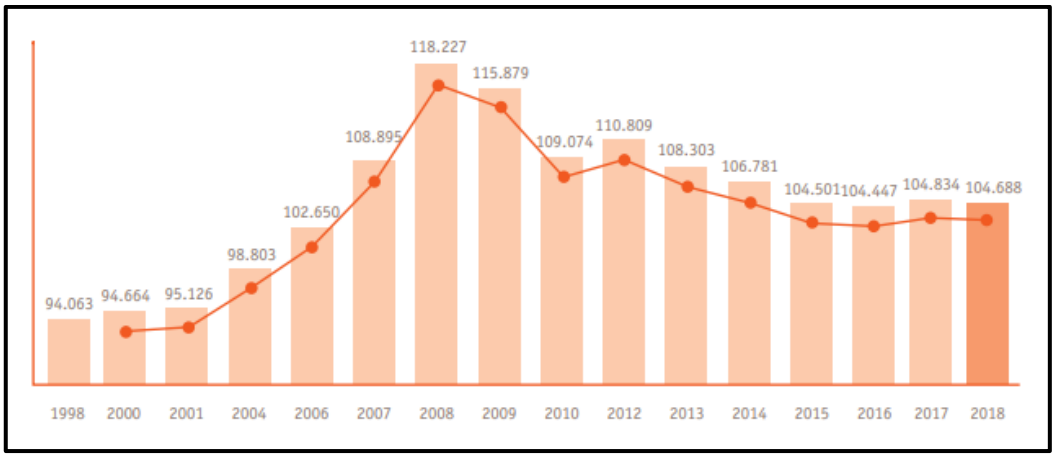

Figura 1. Evolución del número de perros que llegan cada año a refugios de animales. Fuente, Fundación Affinity (2018).

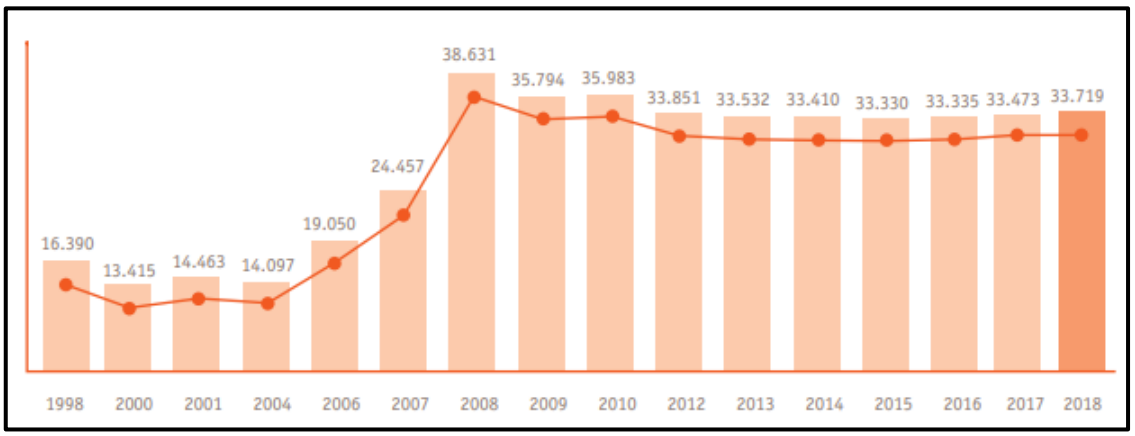

Figura 2. Evolución del número de gatos que llegan cada año a refugios de animales. Fuente, Fundacion Affinity (2018).

En el estudio se logró visualizar que cuando un perro es recogido por un refugio tiene distintos destinos, de los cuales uno de los que sobresalen es que son devueltos.

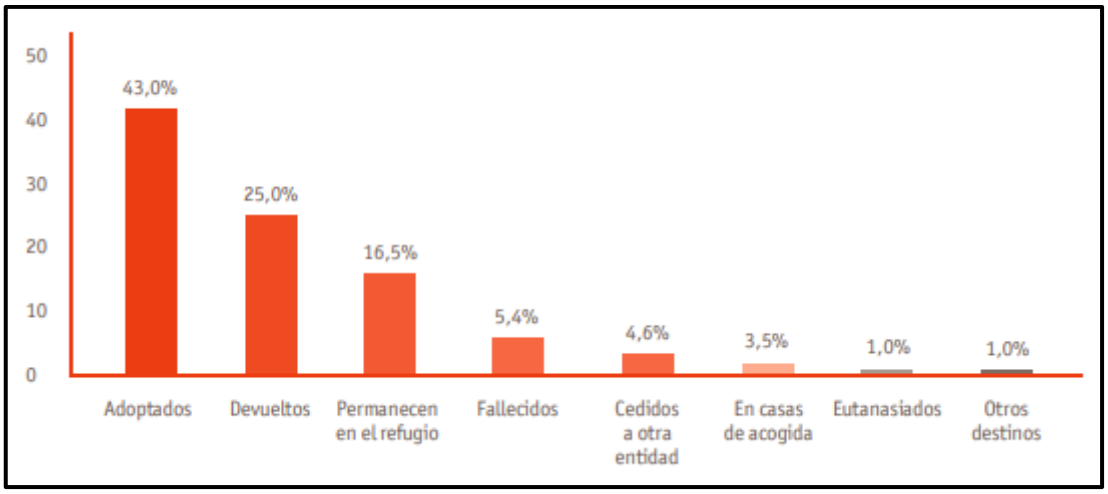

Figura 3. Destino de los perros que llegan a un refugio o protectora de animales. Fuente, Fundación Affinity (2018).

En el estudio se logró visualizar que cuando un gato es recogido por un refugio tiene distintos destinos, donde la probabilidad de que sean devueltos a sus dueños es muy baja

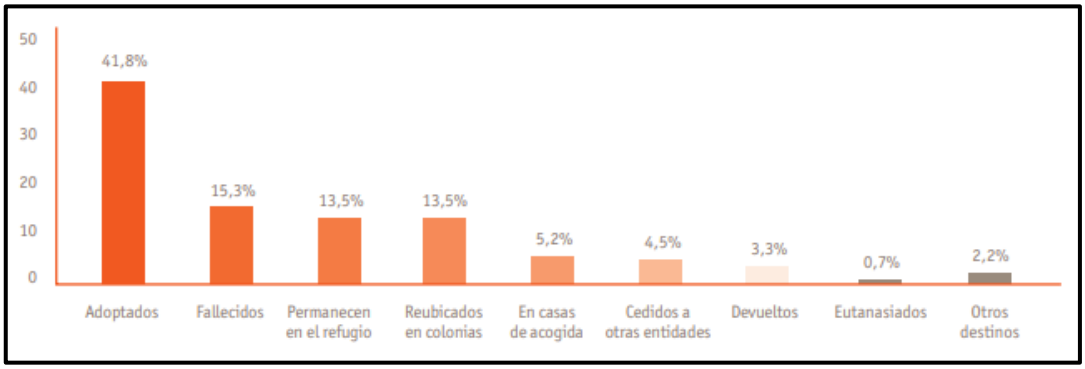

Figura 4. Destino de los gatos que llegan a un refugio o protectora de animales. Fuente, Fundación Affinity (2018)

En la siguiente imagen se muestra el porcentaje de gatos y perros que han sido devueltos a sus dueños luego de haberse perdido.

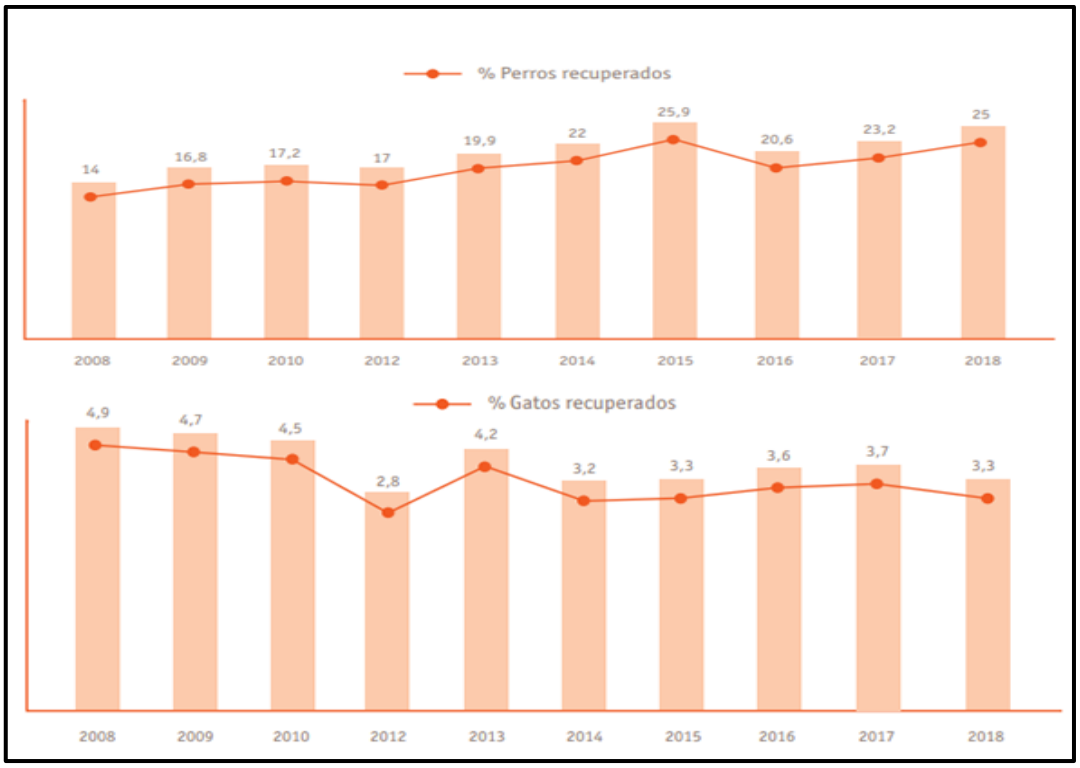

Figura 5. Evolución del porcentaje de perros y gatos que llegan a un refugio de animales y son recuperados por sus dueños. Fuente, Fundación Affinity (2018)

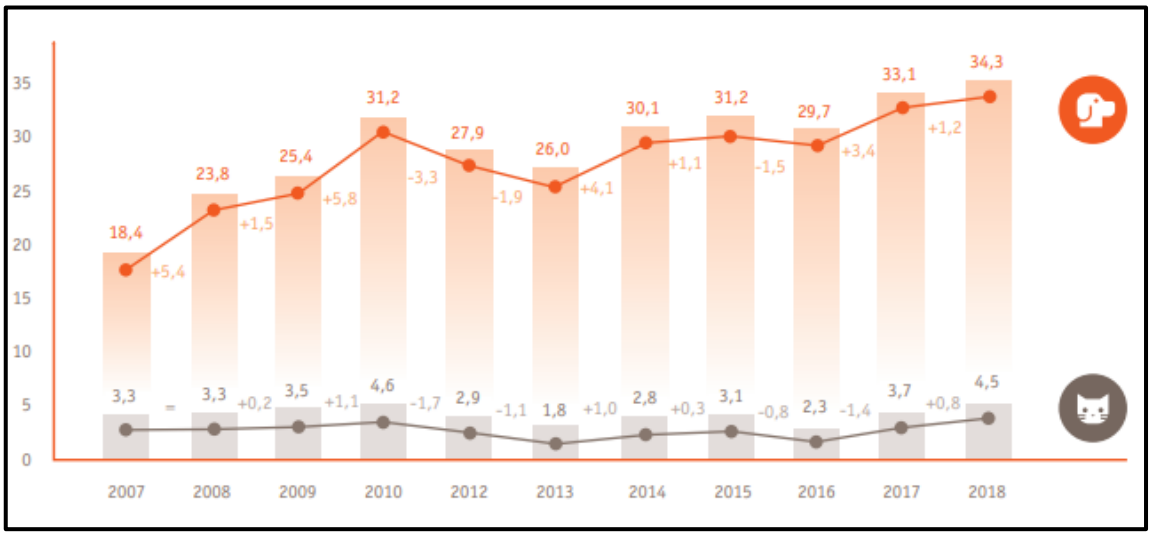

Figura 6. Porcentajes de perros y gatos identificados que llegan a un refugio para animales desde el 2007. Fuente, Fundación Affinity (2018).

Algunos de los métodos o formas para poder combatir el problema de que las mascotas se pierdan y no vuelvan con sus dueños se publicó en una noticia donde utilizaban un código QR colgado en el collar de cada mascota, la dificultad que tendría esta propuesta es que dicho collar se puede perder y en algunas ocasiones

el costo no puede ser afrontado por sus dueños, ya que el precio oscila entre 35 y 50 soles.

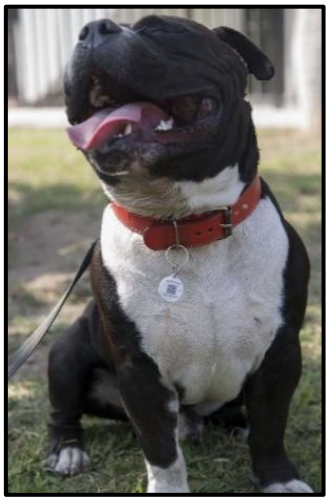

Figura 7. Collar localizador con código QR. Fuente, Andina.pe.

Un método muy parecido al del código QR fue la implantación de un micro chip en las mascotas, donde estás van incrustadas debajo de la piel, cada micro chip tiene un único número identificador a nivel mundial que solo puede ser leído por un lector especial, la deficiencia de este método es que si no se llegará a tener este lector va ser casi imposible poder saber a quién pertenece dicha mascota o saber el número del microchip para luego poder verificar los datos del dueño en la página web petid donde ya está previamente registrado.

Figura 8. Micro chip y lector de micro chip.

(Das, 2013) Otro método alternativo de los que se logró investigar que ayuden a encontrar a las mascotas perdidas, se encontró un aplicativo móvil Finding Rover que es utilizado para encontrar perros perdidos en Estados Unidos. Está idea fue propuesta por John Polimeno y desarrollada por su socio del centro de desarrollo de software de la universidad de Utah, los programadores desarrollaron un algoritmo llamado pet match que utiliza aprendizaje automático para detectar características que diferencien de un perro a otro. Un agregado a esta plataforma es que es muy parecida a la de una red social donde se publica fotos de mascotas que se han perdido y se notifica si algún usuario ha encontrado dicha mascota o inclusive se publican consejos y cuidados de mascotas.

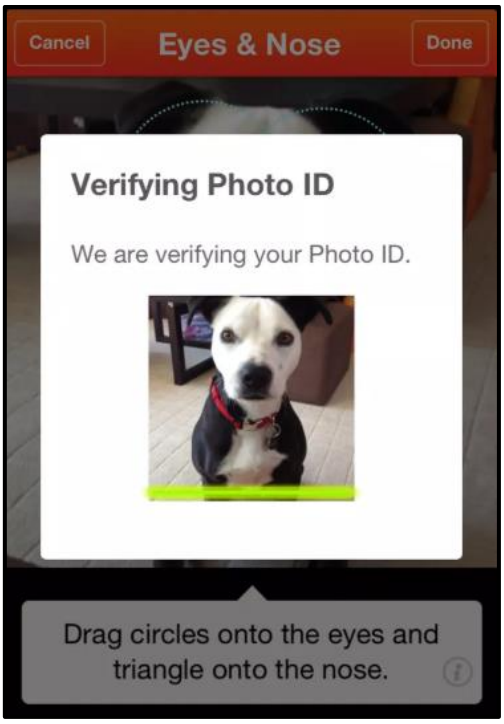

Figura 9. Interfaz del aplicativo móvil Finding Rover.

1.2 Trabajos previos.

Picazo Óscar (2018), *redes neuronales convolucionales profundas para el reconocimiento de emociones en imágenes*, de la universidad politécnica de Madrid, España, realizó una comparación de diferentes modelos populares ya entrenados, pues determinó así que modelo sería el óptimo para realizar modificaciones e inicio el entrenamiento para el reconocimiento de emociones. Luego de haber realizado todas las pruebas entre los modelos de AlexNet, GoogLeNet, VGG, ResNet-101, concluyó que el modelo VGG tiene un mejor resultado entre todas las antes mencionadas, y GoogLeNet teniendo el peor resultado de todos. Aunque VGG no en todas las clases (sorprendido, asustado, asqueado, feliz, triste, enfadado, neutral, media) logró un buen resultado, en las clases de “sorprendido” Res-Net101 obtuvo el mejor resultado y en la clase “feliz” GoogLeNet obtuvo el mejor resultado. En aquel trabajo se concluyó que para obtener una mejora en resultado es recomendable tener una buena resolución de las imágenes y que sean de las mismas dimensiones que los modelos entrenados, además que cambiar el color de la imagen a escalas de grises facilitan mucho para las redes que ha tenido datos en este mismo color.

Ascarza Franco (2018), *segmentación automática de textos mediante redes neuronales convolucionales en imágenes documentos históricos,* de la Universidad Católica del Perú, Perú, realizó la investigación de cómo realizar la segmentación de manera automática en textos, partiendo desde la problemática de que la mayoría de textos son afectados por el tiempo, teniendo como consecuencia que el papel se deteriore o la tinta se corra siendo así difícil poder entender el contenido. El autor propuso desarrollar un modelo basado en una cnn profunda para fragmentar imágenes de documentos históricos. Para realizar la segmentación trabajó con 4 clases (página, texto, decoración y comentarios), luego de implementar la red neuronal convolucional los resultados que obtuvo fueron de que los parches que han sido de tipo decoración y comentario han sido clasificados con menor precisión por el modelo implementado e inclusive los parches de tipo comentario han sido en su mayoría confundidos por los parches de tipo página.

Durán Jaime (2017), *Redes neuronales convolucionales en R,* de la Universidad de Sevilla, España, realizó la investigación para poder reconocer dígitos de caracteres escritos a mano entre 0 al 9, en su trabajo utilizó la base de datos MNIST que contiene sesenta mil caracteres para entrenamiento y diez mil para comprobar, lo que buscó es tener como mínimo un 95% de acierto. Como efecto de la investigación se concluyó que las redes neuronales en especial las redes neuronales convolucionales tiene un alto porcentaje de precisión a pesar de ser una red relativamente fácil de implementar, con 1 hora aproximada de entrenamiento en un computador de gama media con un total de diez mil imágenes de entrenamiento se obtuvo un 96% de acierto, tomando en cuenta que cinco mil de las imágenes estaban distorsionadas.

Martínez Javier (2018), r*econocimiento de imágenes mediante redes neuronales convolucionales,* de la universidad politécnica de Madrid, España, realizó la investigación para poder reconocer y clasificar especies de ballenas jorobadas basando en la cola del animal, utilizó las bibliotecas de tensorflow y keras para implementar la red neuronal convolucional, en aquel trabajo se llegó a la conclusión que el rendimiento no radica necesariamente una buena arquitectura sino que también influye mucho la base de datos de imágenes, que sea balanceada y que tenga una gran cantidad

Mukai Nobuhiko, Zhang Yulong, Chang Youngha (2018), *Pet face detection*, de la Universidad de la ciudad de Tokyo, Japón, se realizó la investigación de cómo realizar la detección de rostros de gatos y perros, el trabajo parte de trabajos previos de detección de rostros en humanos, donde tiene como referencia el algoritmo adaboost, pero no aseguró de que se obtuvo los mismos resultados con las mascotas, como característica adicional se utilizó histograma de gradientes orientados (HOG). Se utilizó 4 formas de detección, clasificar la cara completa con haar-like y HOG y se clasificó la cara por partes con haar-like y HOG. El resultado que se obtuvo de los clasificadores construidos con Haar-Like y HOG indica que son muy útiles para la detección de rostros de mascotas, sin embargo, en el estudio solo se trabaja con 3 tipos de perros de los 120 tipos que existen.

Santosh Kumarl, Shrikant Tiware, Sanjay Kumar Singh (2015), *Face Recognition for Cattle,* El problema central que se afrontó en este artículo es la pérdida o robo del ganado vacuno, se están desarrollando estándares globales para el reconocimiento, registro y trazabilidad del ganado, pero sin embargo, el ganado perdido o intercambiado es difícil de detectar, e inclusive hay los reclamos de falsos y la reasignación de ganado en los mataderos son problemas que estuvieron en todo el mundo. Los enfoques anteriores de reconocimiento de ganado tienen sus propios límites y no pueden proporcionar el nivel de seguridad requerido para el ganado vacuno. En aquel documento, se intentó minimizar los problemas mencionados anteriormente mediante descriptores de reconocimiento facial automático de ganado. El algoritmo de resolución múltiple propuesto extrae la función a través de la función robusta acelerada (SURF) y los patrones binarios locales (LBP) de diferentes niveles de la pirámide gaussiana. Los descriptores de características obtenidos en cada unidad de área de nivel gaussiano se combinaron utilizando técnicas de fusión de reglas de suma ponderada. El algoritmo propuesto arrojó una precisión de identificación de rango del 92.5% en una base de datos de 1200 imágenes de la cara de ganado (10 imágenes por vaca de 120 vacas distintas). Por lo tanto, se llegó a la conclusión en este artículo que la identificación del ganado en función de su rostro se puede utilizar para reconocer el ganado y negar la noción de que todo el ganado se parece.

Vizcaya Cárdenas, Flores Albino, Landassuri Moreno, & Lazcano Salas (2007), *Desempeño de una Red Neuronal Convolucional para Clasificación de Señales de Tránsito Vehicular,* de la Universidad Autónoma del Estado de México, México, se realizó la investigación de medir el desempeño de una cnn para la clasificación de señales de tránsito utilizando un modelo que tenga pocas capas de convolucion y aun así los resultados no se vean afectados, la tasa de error más baja que se obtuvo en la clasificación fue de 8.87% con 1400 épocas, en un tiempo de entrenamiento de 3.16 horas, lo que se llegó a la conclusión que por cada 100 imágenes de tránsito que se le presentó a la cnn se esperó que aproximadamente 91 de ella la clasifique de manera correcta. También indicó que era difícil poder comparar sus resultados con otros trabajos debido a la cantidad y tipo de imágenes que se manejó. Llega a la conclusión también que para obtener mejores resultados a la cnn se le debe ingresar datos en gran cantidad y variados.

Kyaw, Nwe, & Yee (2019), *Pest Classification and Pesticide Recommendation System*, de la University of Computer Studies, Maubin, Myanmar, se afrontó la problemática que Myanmar es un país agrícola que afronta problemas de plagas y que para los agricultores es difícil poder identificar a simple vista y que pesticida utilizar, se propone como solución implementar un sistema de reconocimiento de plagas e indicar que tipo de pesticida sería el más adecuado para utilizar. El modelo que es utilizado es GoogLeNet y se trabajó con un dataset de 1265 imágenes de las 4 plagas que se desea clasificar. Cuando el sistema detecta el tipo de plaga lo que mostro al usuario es el tipo de plaga, el nombre, la familia a la que pertenece, su nombre científico y el pesticida que se debe utilizar.

Jiménez Moreno, Martínez Baquero, & Rodríguez Umaña, (2018), *Sistema automático de clasificación de peces,* de la Universidad Distrital Francisco José De Caldas, Colombia, se realizó el trabajo de investigación para clasificar 2 tipos de peces (tilapia y mojarra) en un ambiente controlado, se utilizó una banda transportadora con color uniforme y luz natural de día donde se colocaron los peces, para la adquisición de las imágenes se utilizó una cámara web convencional con una resolución de 640 x 480 que luego fueron redimensionadas a 180 x 180 para que se reduzca el costo computacional al momento de entrenar. Los resultados que se obtuvo en este trabajo fueron del 100%, este resultado se debe

a que las fotos obtenidas no tenían mucho ruido y fueron tomadas en diferentes ángulos. La cantidad de imágenes que se utilizó fueron de 100 por cada clase, dividiendo así 80% para entrenamiento y 20% para validación.

Quintero, Merchán, Cornejo, & Sanchez, (2018), *Uso de redes neuronales convolucionales para el reconocimiento automático de imágenes de macroinvertebrados para el biomonitoreo participativo*, de la Universidad Tecnológica de Panamá, Panamá, este trabajo tiene como objetivo desarrollar un sistema capaz de reconocer dos familias de macroinvertebrados (Calopterygidae y Heptageniidae) mediante el uso de imágenes, para este trabajo se utilizó el algoritmo de Inception V3, con una base de datos de 200 imágenes organizadas en carpetas por familia y en formato jpg con un tamaño de 300 x 300 pixeles, luego estas imágenes fueron redimensionadas a 224 x 224 pixeles y divididas en 70% para entrenamiento y 30% para validación, la selección de las imágenes se hicieron de manera aleatoria. Para el proceso de entrenamiento se utilizó transferencia de aprendizaje y se obtuvo una precisión de 90%.

1.3 Teorías relacionadas al tema

Machine Learnig: Es una disciplina científica de la inteligencia artificial que implementa sistemas artificialmente inteligentes, basadas en una serie de algoritmos que logran este fin. Dentro de los tipos más conocidos tenemos:

Aprendizaje supervisado: Consiste en entrenar un algoritmo dándole datos de entrada (características) y datos de salida (resultado) logrando así poder obtener una relación entre las variables de entrada y las variables de salida. Se denomina supervisado porque se le da los datos de entrada y salida de lo que se quiere clasificar o predecir.

Aprendizaje no supervisado: Consiste en entrenar un algoritmo dándole datos de entrada sin necesidad de explicarle que es lo que buscamos obtener, a diferencia del aprendizaje supervisado que se da los datos de salida o de lo que se busca obtener.

Deep Learning: Estás redes imitan las conexiones que tiene el cerebro humano, tiene la base del machine learning, esta disciplina es utilizada cuando se maneja

una gran cantidad de datos porque mientras más datos se tenga es mucho más precisa en las predicciones.

Exactitud: Es el resultado de la fracción de predicciones que el modelo realizó correctamente.

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Precisión: Es la proporción de valores positivos sobre todos los resultados positivos (verdaderos y falsos positivos).

$$\frac{TP}{TP + FP}$$

Sensibilidad: Es la proporción de valores positivos sobre la suma de verdaderos positivos y falsos negativos.

$$\frac{TP}{TP + FN}$$

Tensorflow: E**s** una biblioteca de código abierto que se utiliza para construir y entrenar redes neuronales, fue creada por google, tensorflow es utilizado para implementar sistemas deep learning y redes neuronales. Además, que es la más popular entre otras bibliotecas. (Llamas, 2018)

Tabla 1.

Principales plataformas de aprendizaje profundo

Plataforma	2019	2018
Tensorflow	31.7%	29.9%
Keras	26.6%	22.2%
PyTorch	11.3%	6.4%
Other Deep Learning Tools	5.6%	4.9%
DeepLearning4J	2.5%	3.4%
Apache MXnet	1.7%	1.5%
Microsoft Cognitive Toolkit	1.6%	3.0%

Theano	1.6%	4.9%
Torch	0.9%	1.0%
TFLearn	0.7%	1.1%
Caffe	0.6%	1.5%

Fuente: Recuperado de (Piatetsky, 2019)

Keras: Es una biblioteca que es utilizada para el aprendizaje (implementar redes neuronales), está escrita en python y corre en bibliotecas como thenao, tensorflow. Luego keras fue unido a tensorflow, esta integración permite que el nivel de abstracción al momento de implementar sea mucho mayor.

Python: Es un lenguaje de programación denominado de alto nivel porque es muy fácil de leer, utiliza una sintaxis sencilla. Una de sus características es que es multiparadigma ya que se puede utilizar programación orientada a objetos, programación imperativa y en menor medida programación funcional. (Wikipedia, 2019) Además de que estos dos últimos años python ha sido el lenguaje más popular entre otros lenguajes de programación.

Tabla 2.

Lenguajes de programación más populares

Plataforma	2019	2018
Python	65.8%	65.6%
R Language	46.6%	48.5%
SQL Language	32.8%	39.6%
Java	12.4%	15.1%
Unix shell/awk	7.9%	9.2%
C/C++	7.1%	6.8%
Javascript	6.8%	--
Other programming and data languages	5.7%	6.9%
Scala	3.5%	5.9%
Julia	1.7%	0.7%
Perl	1.3%	1.0%
Lisp	0.4%	0.3%

Fuente: Recuperado de (Piatetsky, 2019)

Cuda: Fue desarrollada por nvidia, explota las ventajas de la gpu al momento de procesar imágenes utilizando paralelismo en sus instrucciones, ayudando así a agilizar la etapa de procesamiento de imágenes.

Redes neuronales convolucionales: Nacen con la necesidad de procesar imágenes de una manera efectiva y eficiente, también se utiliza para texto. Su estructura es similar a las redes neuronales ya que cuentan con una capa de entrada, capas ocultas y una capa de salida. En las capas ocultas se realiza la operación de pooling o agrupación la cual sirve para reducir el tamaño de la imagen que estamos procesando para que sea mucho más fácil procesar para el computador, la otra operación se denomina convolución, es donde se aplica varios filtros a las imágenes para poder detectar características ya sea líneas, bordes, etc.

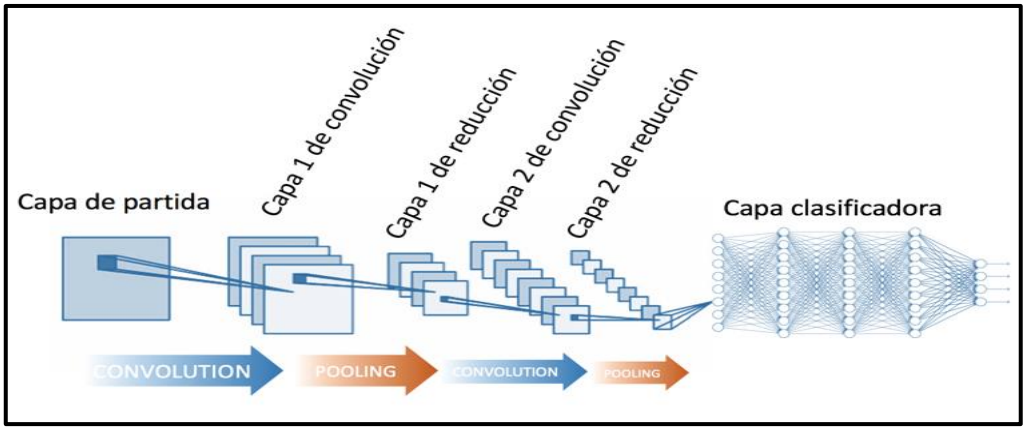

Figura 10. Arquitectura de una red neuronal convolucional.

Convolución: La convolución es como tener una red neuronal pequeña que analiza ciertos sectores de la imagen y no toda la imagen completa, luego que hace este análisis se obtiene una nueva imagen de menor ancho y altura, pero de mayor profundidad, la profundidad aumenta según la cantidad de filtros que se aplique a la imagen en cada una de la convolución.

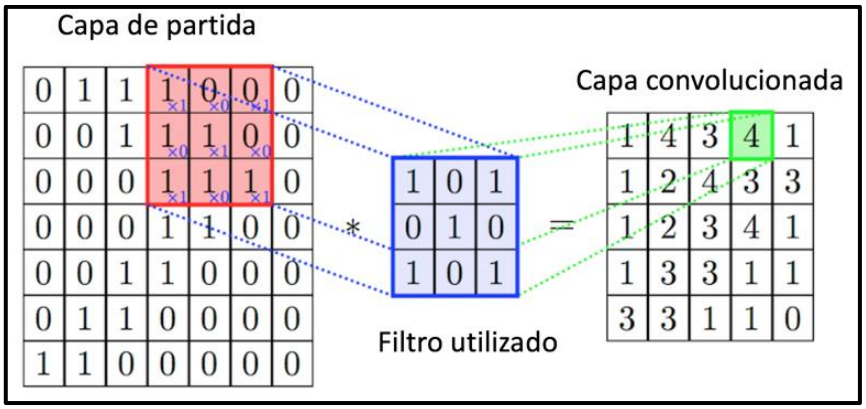

Figura 11. Proceso de convolución de una red neuronal convolucional.

Pooling: Es la operación de reducción de tamaño de la imagen para que sea mucho más sencilla de procesar. Esta operación se realiza a través de dos formas, average pooling que saca el promedio o max pooling que obtiene el valor máximo de una región de la imagen.

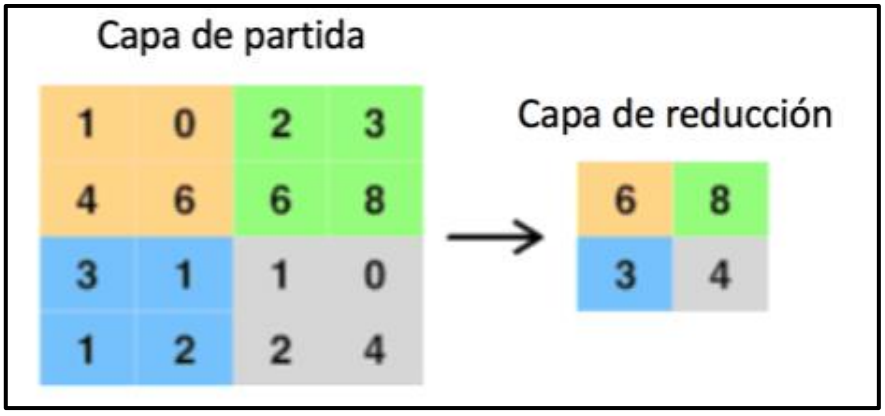

Figura 12. Proceso de pooling en una red neuronal convolucional.

Función softmax: Se usa normalmente en redes neuronales para clasificar datos. Nos da un porcentaje que sumado todos sus valores es 1.

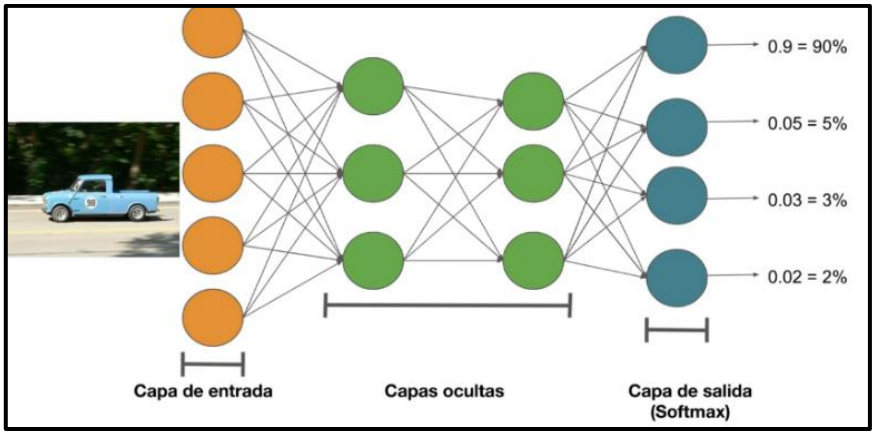

Figura 13. Arquitectura de la función de activación softmax.

Matriz de confusión: La matriz de confusión es una herramienta que se utiliza para saber qué tan efectivo es el modelo de clasificación teniendo en cuenta los errores y aciertos de las pruebas que se realizan, esta herramienta nos ayuda a decidir si el modelo entrenado está clasificando bien las clases. (Ecured, s.f.)

Tabla 3

Modelo de matriz de confusión

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos positivos (VP)	Falsos negativos (FN)
	Negativos	Falsos positivos (FP)	Verdaderos negativos (VN)

Fuente: Elaboración propia.

Scikit-learn: Es una biblioteca de código abierto para aprendizaje supervisado y no supervisado en python. Contiene numpy, pandas, scipy, matplotlib y sympy.

NumPy: Es una biblioteca que se utiliza para trabajar con matrices n dimensional o vectores.

Matplotlib: Es una biblioteca que se utiliza para generar gráficos 2d.

SciPy: Es una biblioteca que se utiliza para computación científica, contiene módulos para integración, funciones especiales, optimización, álgebra lineal, interpolación, procesamiento de señales y de imagen.

1.4 Formulación del Problema

¿Qué tan efectivas son las redes neuronales convolucionales para identificar gatos en imágenes digitales?

1.5 Justificación e importancia del estudio

Está investigación fue muy importante porque brindó un aporte a los estudios de reconocimiento de imágenes que utilizan redes neuronales convolucionales, ya que la mayor parte de los trabajos de investigación trabajan con un número determinado de clases y en este trabajo el número de clases incrementa por cada gato.

1.6 Hipótesis

Las redes neuronales convolucionales son muy efectivas para detectar gatos en imágenes digitales.

1.7 Objetivos

1.7.1 Objetivos General

Identificar automáticamente gatos en imágenes digitales usando redes neuronales convolucionales

1.7.2 Objetivos Específicos

Elaborar base de datos con imágenes de gatos.

Determinar cuántas capas va a tener el modelo.

Construir el modelo de la red neuronal convolucional.

Evaluar resultados.

II. MATERIAL Y MÉTODO

2.1 Tipo y Diseño de Investigación

Cuasi experimental

2.2 Población y muestra.

Población: 500 imágenes de entrenamiento, 170 para validación y 165 de prueba.

Muestra: 60 % para entrenamiento, 20% para validación y 20 % para prueba

2.3 Variables, Operacionalización

Variables dependientes:

Identificación de gatos.

Variables independientes:

Redes neuronales convolucionales.

Operacionalización:

Tabla 4

Operacionalización de variable dependiente

Variable Dependiente	Dimensiones	Indicadores	Fórmula	Técnica e instrumento de recolección de datos
Identificación de gatos.	Eficiencia	Exactitud	$\frac{TP+TN}{TP+TN+FP+FN}$	Observación, ficha de observación
		Precisión global	$\frac{suma\ de\ diagonal\ principal}{TP+TN+FP+FN}$	
		Precisión por clase	$\frac{TP}{TP+FP}$	
		Sensibilidad	$\frac{TP}{TP+FN}$	

Fuente: Elaboración propia.

Tabla 5

Operacionalización de variable independiente

Variable Independiente	Dimensiones	Indicadores	Fórmula	Técnica e instrumento de recolección de datos
Redes neuronales convolucionales.	Rendimiento	Tiempo de Entrenamiento	$Hora\ de\ inicio - Hora\ de\ fin$	Observación, ficha de observación

Fuente: Elaboración propia.

2.4 Técnicas e instrumentos de recolección de datos, validez y confiabilidad

Observación:

Se utilizó esta técnica de recolección de datos porque se tuvo que dar seguimiento a los resultados que se obtuvieron de cada prueba de entrenamiento pues así se determinó cuál fue la más eficiente.

Ficha de observación:

La ficha de observación se utilizó porque se necesitó registrar todos los datos de lo observado en las pruebas realizadas además de que la ficha de observación fue la relación entre la hipótesis y los hechos reales.

2.5 Procedimientos de análisis de datos

Basándose en las técnicas de recolección de datos se consideró:

En la primera etapa se adquirió las imágenes con 2 celulares y luego se armó la base de datos con todas las imágenes adquiridas.

En la segunda etapa luego de haber hecho las pruebas y basándose en los artículos leídos se determinó la estructura del modelo.

En la tercera etapa se implementó la red neuronal convolucional.

En la cuarta etapa se evaluó los resultados.

2.6 Criterios éticos

Derechos de autor: Todo material que se usó para el proceso de tema de tesis estará referenciado y citado con sus respectivos autores.

Confidencialidad: Se aseguró la protección de identidad de sus fuentes, como también de las personas que participaron como informantes de la investigación.

2.7 Criterios de Rigor científico.

Fiabilidad: El proyecto de investigación se realizó el cumplimiento de las expectativas mencionadas en su contenido y se implementara con los estándares de desarrollo.

Consistencia: El proyecto de investigación representará material consistente, fiable y certificado por la comunidad científica.

Validez: Los datos alcanzados por el proyecto serán examinados por los ingenieros especialistas en el tema para determinar su autenticidad.

III. RESULTADOS

3.1 Resultados en Tablas y Figuras

El modelo fue entrenado con 100 imágenes y fue validado con 34 imágenes, en la etapa de entrenamiento se determinó utilizar 20 épocas, 500 pasos, 100 pasos de validación, 3 filtros de convolución de 32, 64 y 128, el tamaño de cada filtro es de (3,3), (2,2) y (2,2) con un aprendizaje de 0.0005.

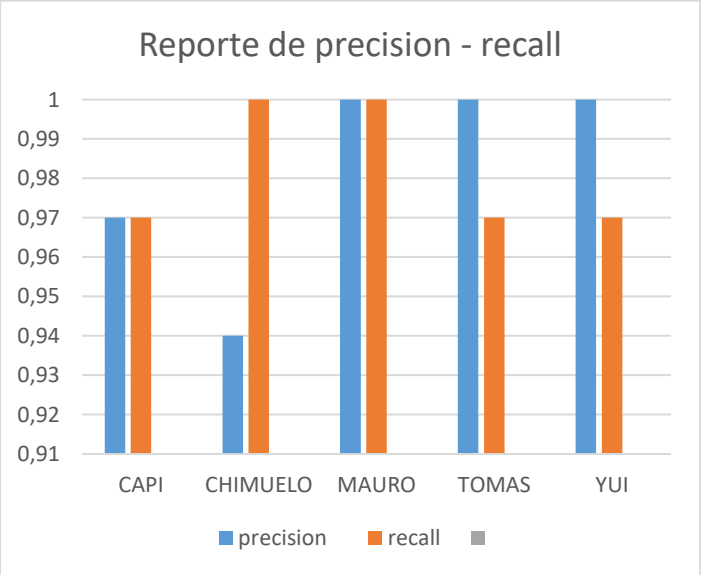

Figura 14. Análisis precisión y recall de las 5 clases de prueba.

Al momento que se realizó las pruebas en la clase 1 (capi) de 33 fotos de prueba, 32 fotos fueron verdaderos positivos, 1 fue un falso negativo confundiendo a la clase 2 por la clase 1 obteniendo así un 97% de sensibilidad (recall) y 1 falso positivo donde se obtuvo así un 97% de precisión.

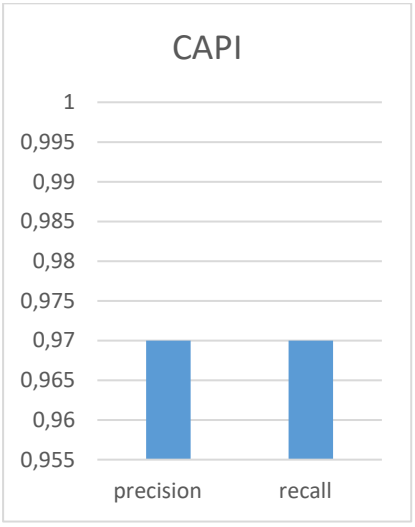

Figura 15. Análisis precisión y recall de la clase capi.

Al momento que se realizó las pruebas en la clase 2 (chimuelo) de 33 fotos de prueba, 33 fotos fueron verdaderos positivos, 0 falsos negativos donde se obtuvo así el mayor porcentaje de sensibilidad (recall) y 2 falsos positivos donde se obtuvo así un 94 % de precisión.

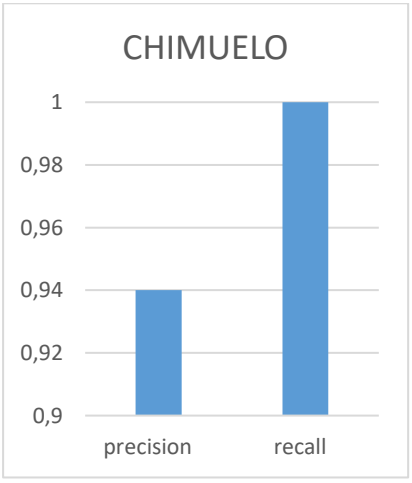

Figura 16. Análisis precisión y recall de la clase chimuelo.

Al momento que se realizó las pruebas en la clase 3 (mauro) de 33 fotos de prueba, 33 fotos fueron verdaderos positivos, 0 falsos negativos donde se obtuvo así el mayor porcentaje de sensibilidad (recall) y 0 falsos positivos donde se obtuvo así el mayor porcentaje de precisión.

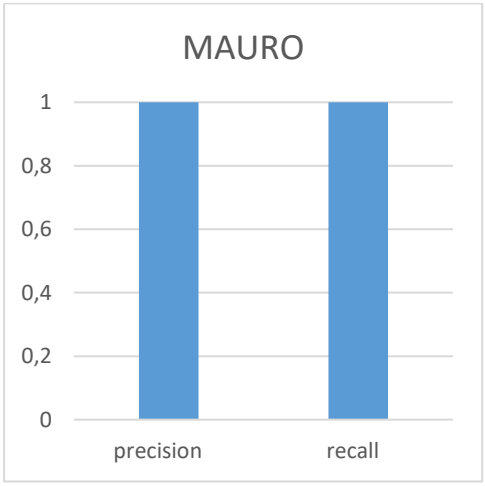

Figura 17. Análisis precisión y recall de la clase mauro.

Al momento que se realizó las pruebas en la clase 4 (tomas) de 33 fotos de prueba, 32 fotos fueron verdaderos positivos, 1 falsos negativos donde se obtuvo así el 97% de sensibilidad (recall) y 0 falsos positivos donde se obtuvo así el mayor porcentaje de precisión.

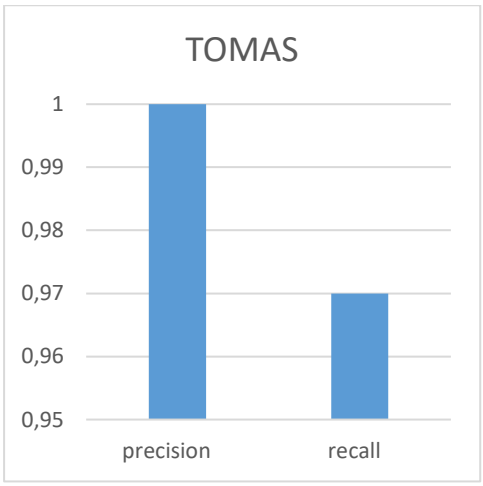

Figura 18. Análisis precisión y recall de la clase tomas.

Al momento que se realizó las pruebas en la clase 5 (yui) de 33 fotos de prueba, 32 fotos fueron verdaderos positivos, 1 falsos negativos donde se obtuvo así el 0.97% de sensibilidad (recall) y 0 falsos positivos donde se obtuvo así el mayor porcentaje de precisión.

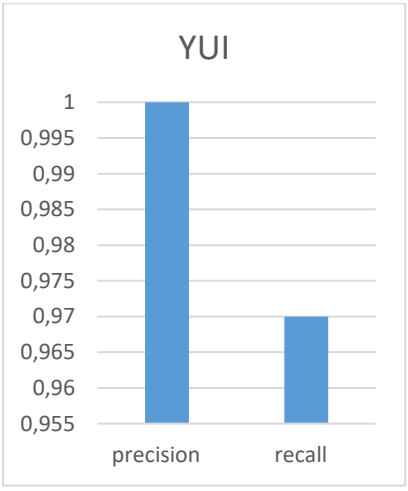

Figura 19. Análisis precisión y recall de la clase yui.

3.2 Discusión de resultados

Tabla 6

Comparación de resultados entre otros modelos trabajados en otros trabajos

TRABAJO	MODELO	TIEMPO	EXACTITUD	CANTIDAD DE FOTOS ENTRENAMIENTO	CANTIDAD DE FOTOS VALIDACIÓN	GPU
IDENTIFICACION AUTOMATICA DE GATOS MEDIANTE RECONOCIMIENTO DE IMÁGENES USANDO REDES NEURONALES CONVOLUCIONALES	Modelo con 3 capas de convolución	1 hora 52 minutos y 17 segundos	98%	500	170	GPU = NVIDIA GeForce GTX 1050 Ti
	Modelo VGG16	2 horas 1 minutos y 17 segundos	13%	500	170	
REDES NEURONALES CONVOLUCIONALES PROFUNDAS PARA EL RECONOCIMIENTO DE EMOCIONES EN IMÁGENES	Modelo VGG16	4 días y 5 horas	51.08%	12 271	3068	GPU = NVIDIA GeForce GTX 750

Fuente: Elaboración propia.

Los datos que se muestran son de pruebas realizadas en este trabajo y también de otro. Comparando los resultados de exactitud del mismo modelo VGG16 se logró llegar a la conclusión que la cantidad de fotos influyó mucho en el resultado al momento de realizar el entrenamiento, ya que al tener una cantidad pequeña de fotos (670) ocurre una sobrecarga de aprendizaje y el modelo empieza a perder exactitud, y eso se ve reflejado el resultado de 13% frente a los 51.08% obtenidos de una cantidad de fotos de 15339.

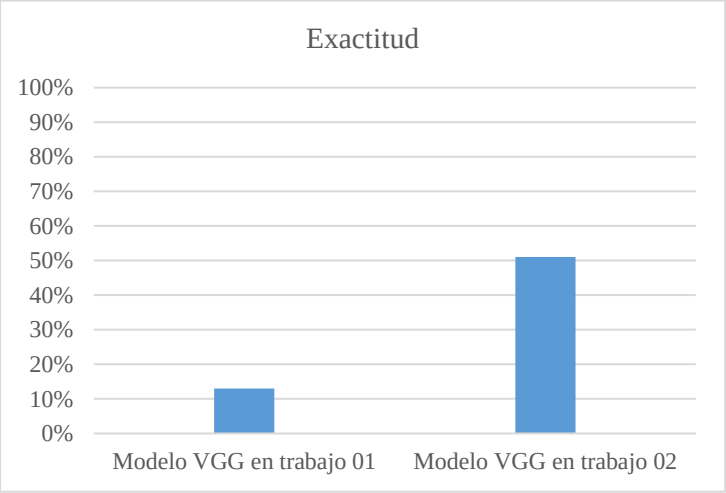

Figura 20. Resultados de exactitud de modelo VGG16 en trabajo 01 y trabajo 02.

En el tiempo de entrenamiento influyó mucho el hardware y la cantidad de fotos por ello se ve reflejado la diferencia abismal del tiempo de entrenamiento de 2 horas 1 minutos y 17 segundos (7277seg.) en 670 fotos frente a los 4 días y 5 horas (363 600seg.) en 15339 fotos.

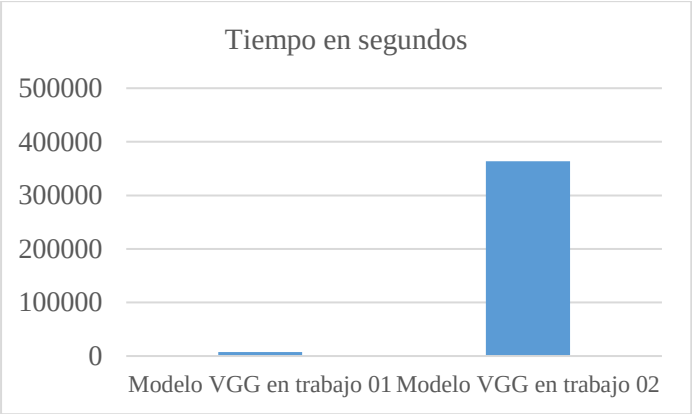

Figura 21. Tiempo de entrenamiento de modelo VGG16 en trabajo 01 y trabajo 02.

Comparando los resultados de exactitud del modelo VGG16 frente al modelo de 3 capas se logró notar una gran diferencia en el porcentaje de exactitud, pero esto no quiere decir que el modelo de 3 capas implementadas sea mejor que el modelo VGG16, solo que el modelo de 3 capas tiene las capas adecuadas para la cantidad de fotos que se entrenó por ello la exactitud es de 98%, en el caso de que se tenga una gran cantidad de fotos y de clases es probable que sea necesario implementar más capas de convolución y maxpooling para así poder obtener más características que ayuden a diferencias una clase de otra.

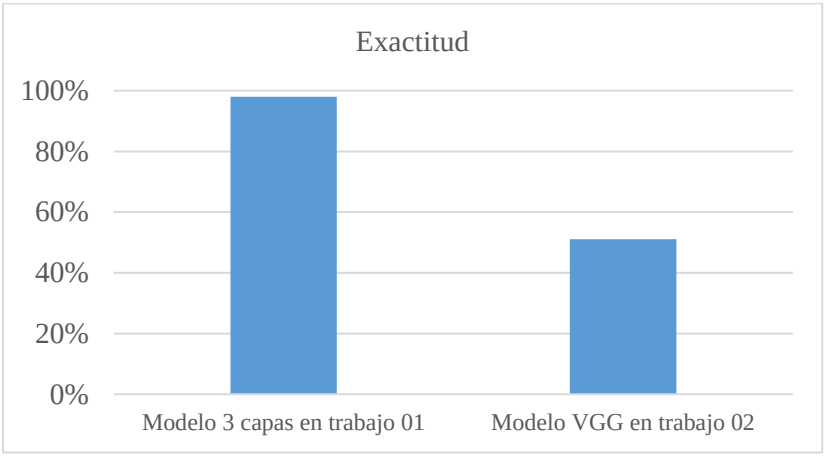

Figura 22. Resultados de exactitud en modelo de 3 capas de convolución en trabajo 01 y modelo VGG16 en trabajo 02.

En el tiempo de entrenamiento influyó mucho el hardware y la cantidad de fotos por ello se ve reflejado la diferencia abismal del tiempo de entrenamiento de 1 hora 52 minutos y 17 segundos (6737seg.) en 670 fotos frente a los 4 días y 5 horas (363 600seg.) en 15339 fotos.

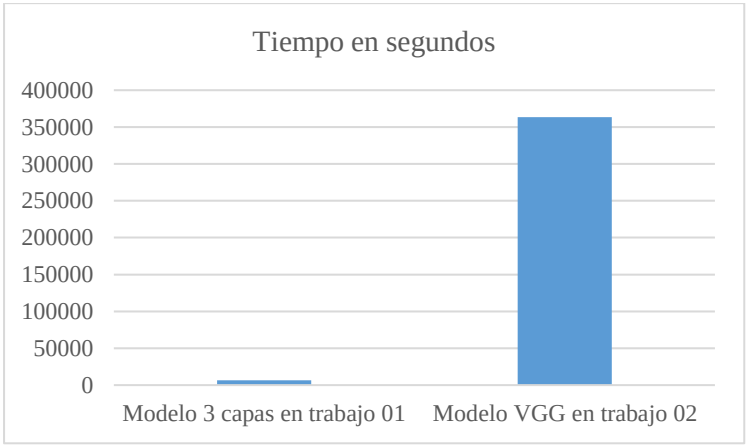

Figura 23. Tiempo de entrenamiento de modelo 3 capas de convolución en trabajo 01 y modelo VGG16 en trabajo 02.

3.3 Aporte práctico

En esta investigación, el método propuesto fue estructurado en 4 pasos. Inició con la adquisición de imágenes de los gatos en un ambiente no controlado. En la segunda parte se armó el dataset con las fotos de los gatos. En el tercer paso se implementó la red neuronal convolucional. En el cuarto paso se probó la red neuronal convolucional.

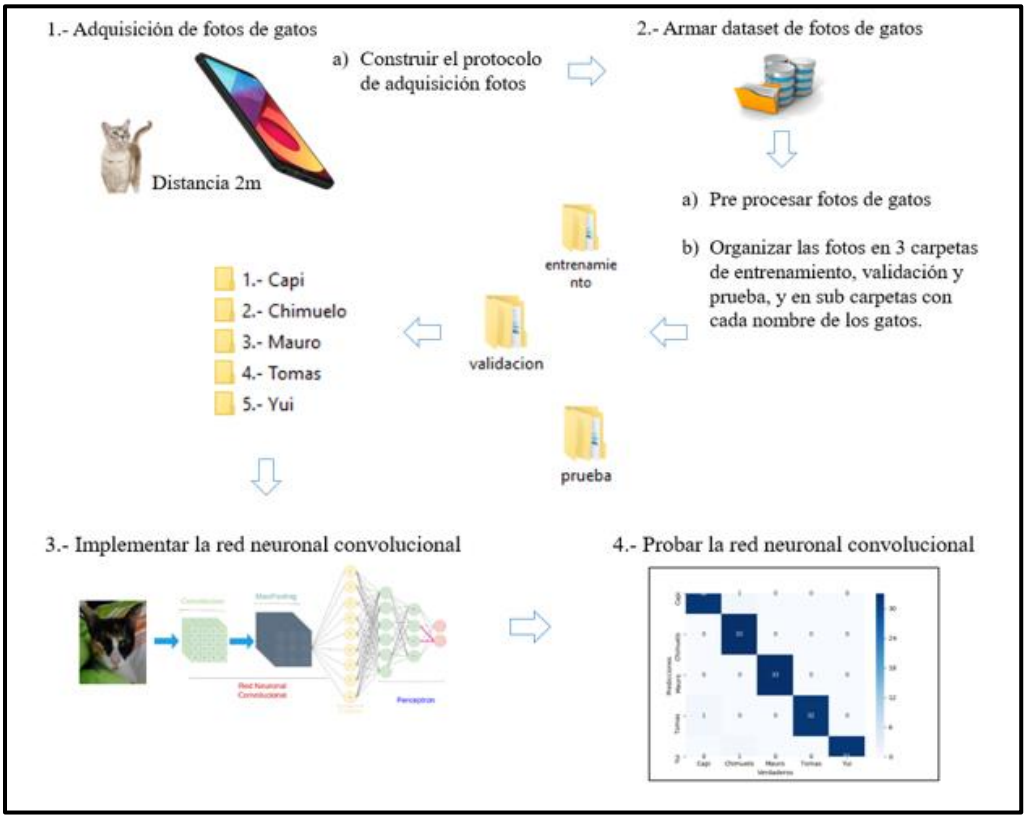

Figura 24. Método general para realizar los objetivos propuestos.

Adquisición de fotos de gatos

Para la adquisición de las fotos se procedió a establecer un protocolo que fue seguido con rigurosidad, la distancia determinada para la toma de las fotos fue de 2 metros, la configuración de la cámara fue en enfoque automático, sin flash, sin filtro y en modo ráfaga con una resolución de 3120 x 4160 y 960 x 1280.

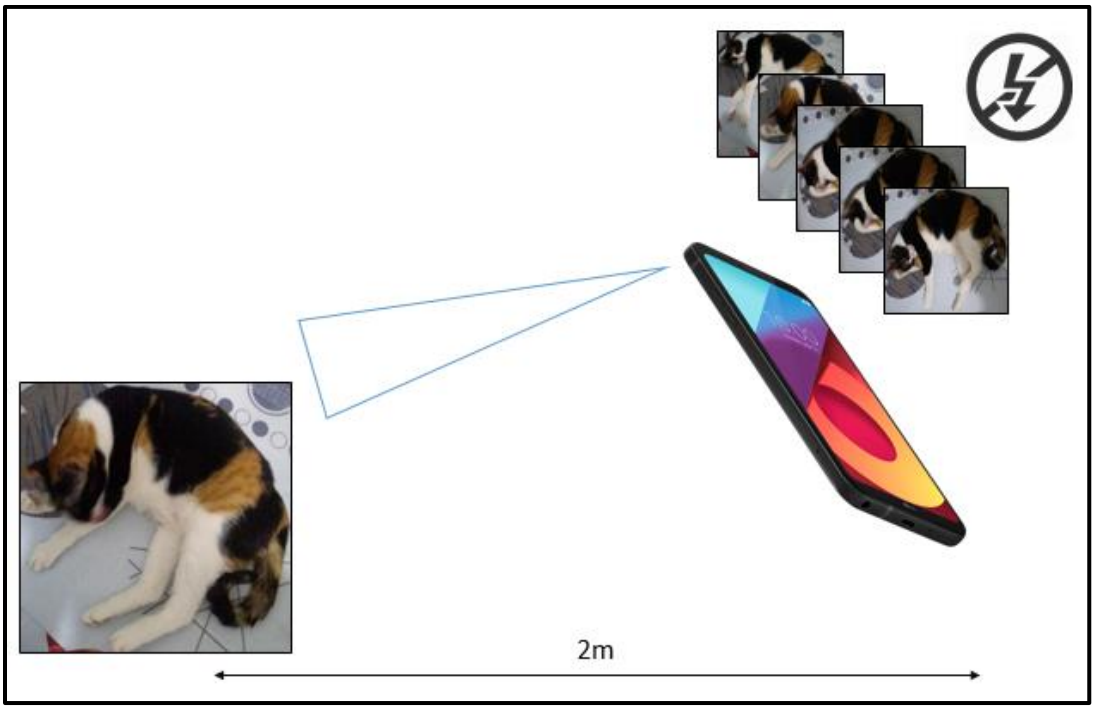

Figura 25. Condiciones para la toma de fotos de gatos.

Las cámaras utilizadas fueron 2 cámaras cotidianas Lg Q6 y Samsung galaxy A50 que tienen las siguientes características.

Tabla 7

Especificaciones de celular LG-Q6

Modelo	Sistema operativo	RAM	Pantalla	Almacenamiento	Cámaras
LG-M700F	Android 7.1.1	3GB	5.5 pulgadas	32 GB	Trasera 13 megapíxeles Frontal 5 megapíxeles

Fuente: Elaboración propia.

Tabla 8

Especificaciones de celular Samsung galaxy A50

Modelo	Sistema operativo	RAM	Pantalla	Almacenamiento	Cámaras
SM-A505G	Android 9	4GB	6.4 pulgadas	512GB de memoria expandible	Tripe cámara trasera 25 MP + 8 MP + 5MP Frontal 25 MP

Fuente: Elaboración propia.

Armar dataset de fotos de gatos

Las fotos adquiridas fueron almacenadas en 3 carpetas con los nombres de entrenamiento, validación y prueba, dentro de ella se almacenó sub carpetas con los nombres de cada gato (capi, chimuelo, tomas, mauro y yui).

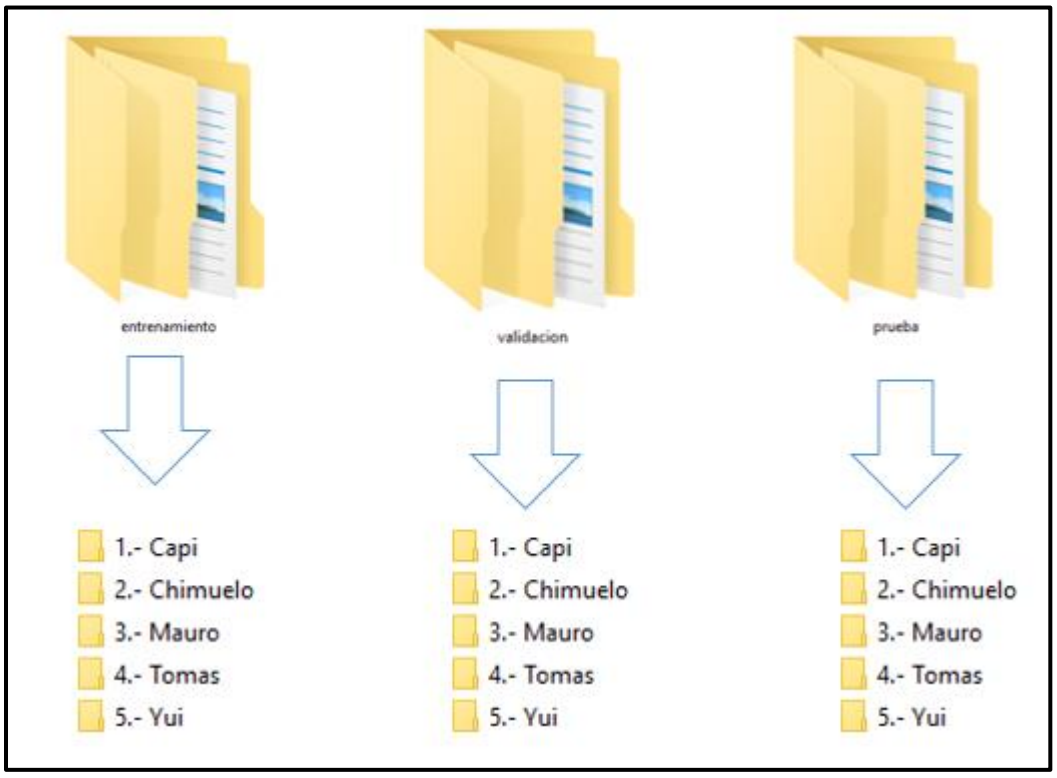

Figura 26. Estructura general del almacenamiento del dataset de fotos de gatos.

Para aumentar el dataset también se realizó el reconocimiento de rostros de gatos y se procedió a recortar. Para todo este procedimiento de detección de rostro se utilizó el algoritmo de haar cascade de detención de rostro de gatos.

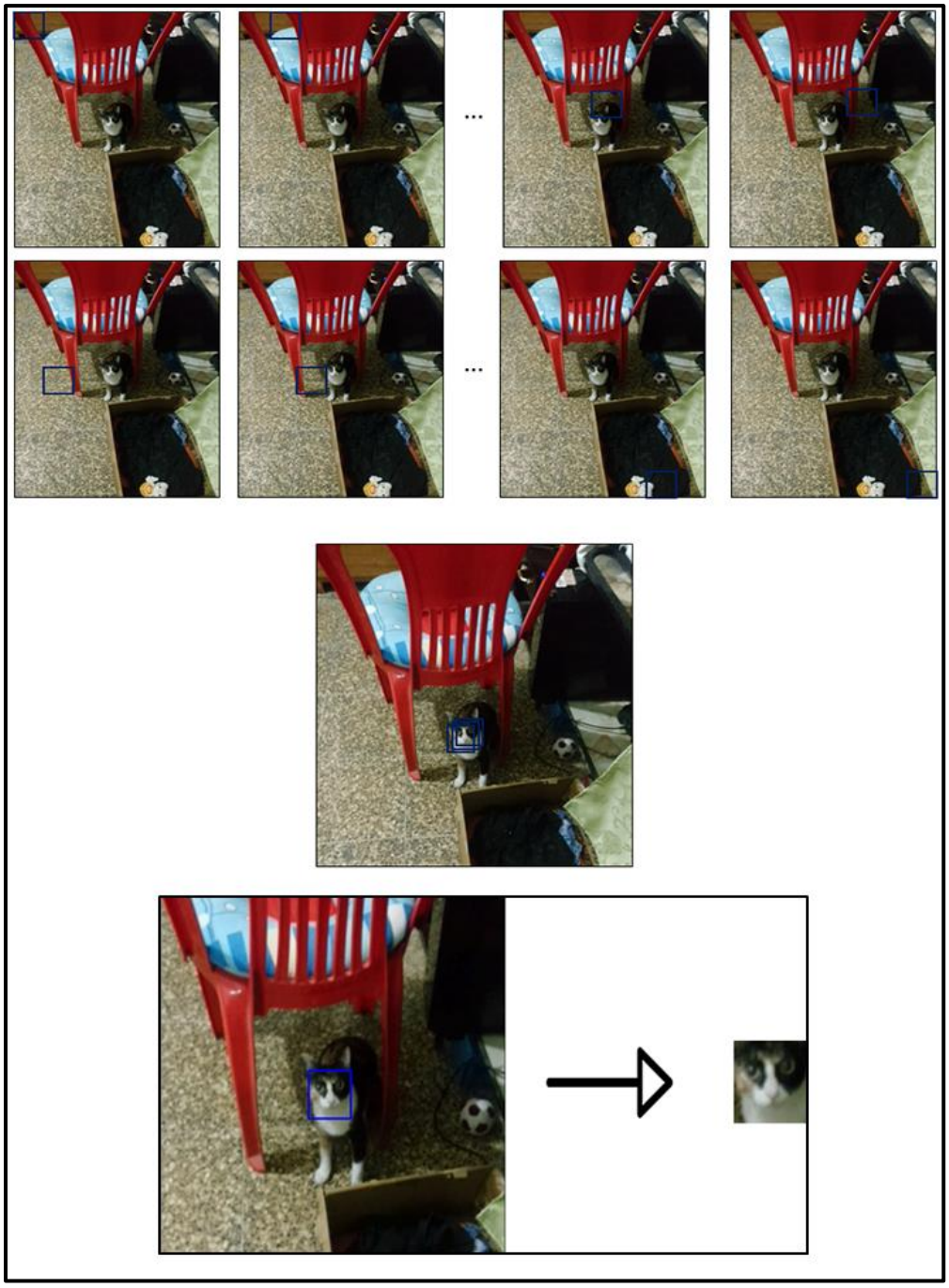

Figura 27. Detección de rostro usando haar cascade para luego recortar.

Luego que ya se tuvo los gatos organizados en sus respectivas carpetas y los rostros recortados también almacenados, se procedió a recortar el área de interés de cada gato.

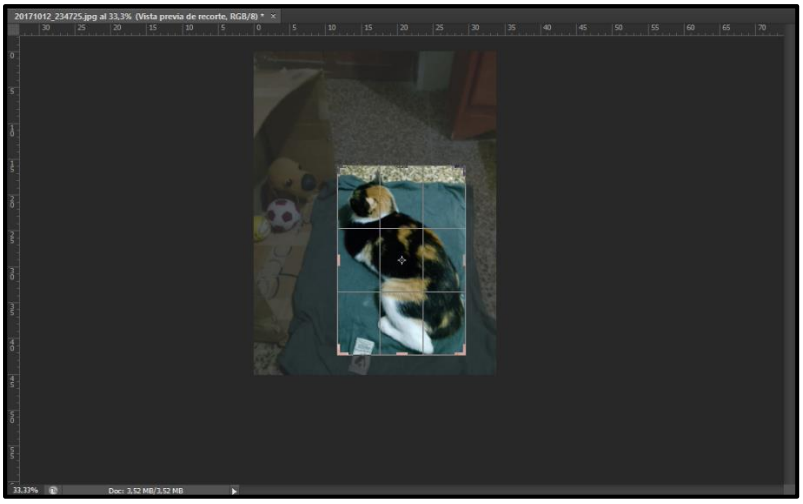

Figura 28. Recorte del área de interés de la foto de un gato.

Las imágenes fueron distribuidas en 60% para entrenamiento, 20% para validación y 20% para prueba. En la carpeta de entrenamiento se almacenó un total de 500 fotos, divididas en 100 fotos por clase, en la carpeta de validación se almacenó 170 fotos divididas en 34 fotos por clases y en la carpeta de prueba se almacenó 165 fotos divididas en 33 fotos por clases.

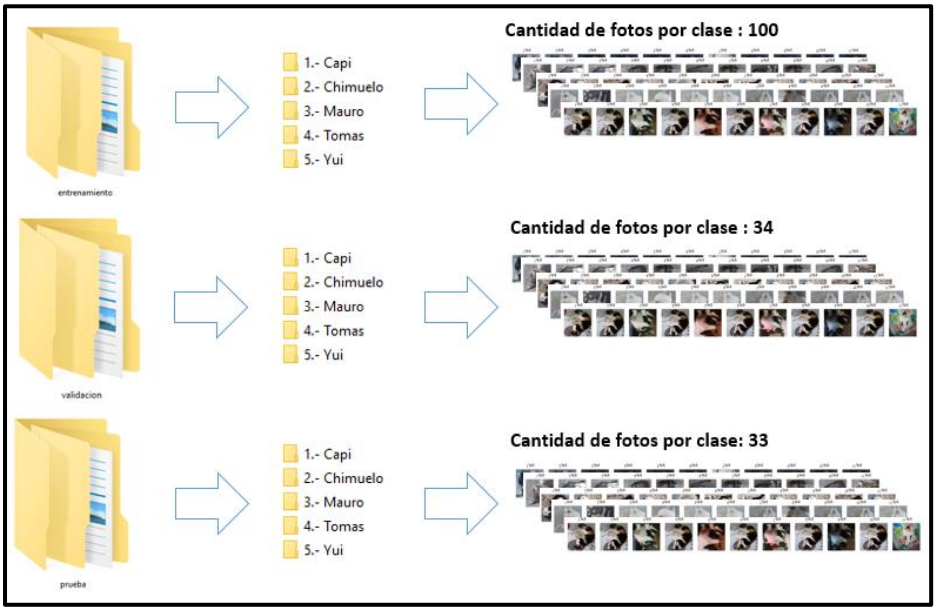

Figura 29. Distribución de fotos de gatos en dataset.

Luego de que se almacenó las imágenes se realizó la redimensión de todas las imágenes en dimensiones de 224x224. Se eligió esta dimensión porque es la dimensión del modelo pre entrenado VGG16 que según los artículos leídos es la que obtuvo mejores resultados.

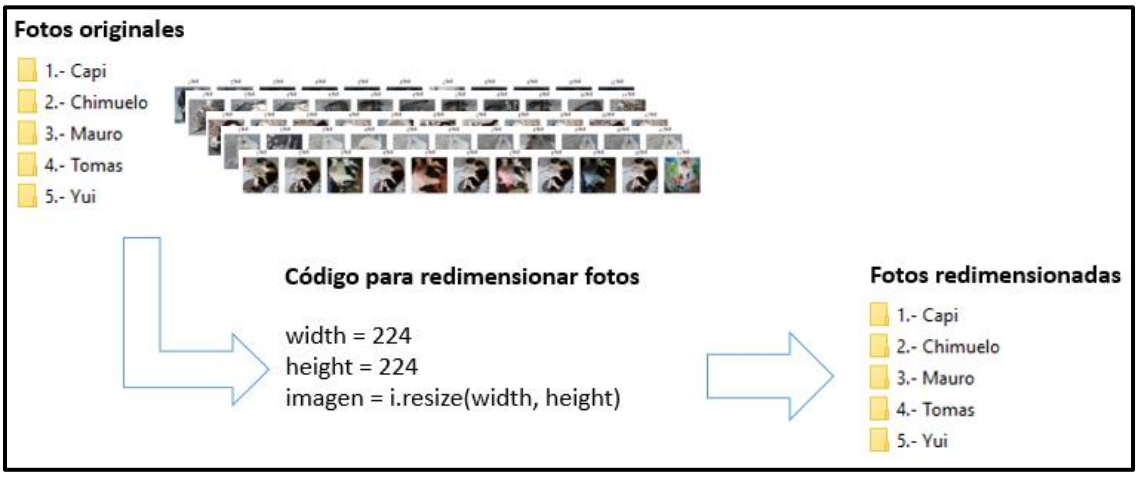

Figura 30. Modelo de redimensión de imágenes de gatos.

Fórmula para redimensionar imagen:

$$dsize = Size(round(fx * src.cols), round(fy * src.rows))$$

Donde:

dsize= es la imagen de salida.

src= es la imagen de entrada.

fx = factor de escala a lo largo del eje horizontal.

fy = factor de escala a lo largo del eje vertical.

Implementar la red neuronal convolucional

Antes de la implementación de la red neuronal convolucional que se utilizó para este trabajo, se procedió a realizar algunas pruebas para determinar la estructura de la red.

$$Y = f(x)$$

Donde:

X = es la entrada.

Y = es la predicción.

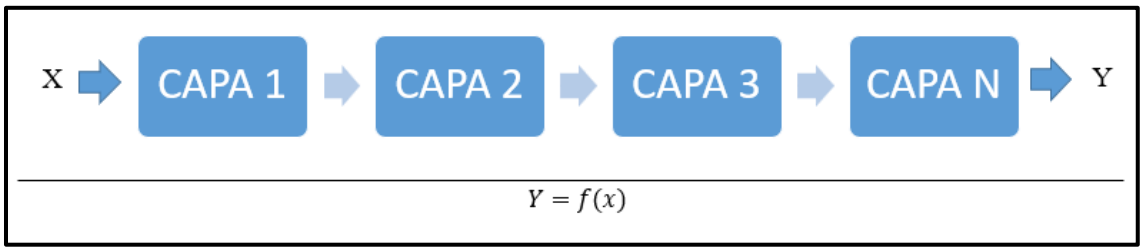

Figura 31. Representación gráfica de la ecuación.

$y1 = Capa_1(X)$

$y2 = Capa_2(y1)$

$y3 = Capa_3(y2)$

$Y = Capa_n(y3)$

$Y = Capa_n(Capa_3(Capa_2(Capa_1(X))))$

Cada capa se encarga de buscar patrones, estos patrones que se aprenden son conocidos como pesos.

Determinar cuántas capas va a tener el modelo.

Para determinar la cantidad de capas que tuvo nuestro modelo se implementaron 3 modelos y se utilizó transferencia de aprendizaje utilizando el modelo de VGG16 que ya está implementado. Los modelos creados fueron una con 2 capas de convolución y 2 capas de maxpooling, seguido de una capa totalmente conectada y una capa de salida.

Tabla 9

Estructura de primer modelo entrenado.

Capa	Salida	Parámetros
conv2d_1 (Conv2D)	(None, 224, 224, 32)	896
max_pooling2d_1(MaxPooling2)	(None, 112, 112, 32)	0
conv2d_2 (Conv2D)	(None, 112, 112, 64)	8256

max_pooling2d_2(MaxPooling2)	(None, 56, 56, 64)	0
flatten_1 (Flatten)	(None, 200704)	0
dense_1 (Dense)	(None, 256)	51380480
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 5)	1285

Fuente: Elaboración propia.

Otra con 3 capas de convolución y 3 capas de maxpooling, seguido de una capa totalmente conectada y una capa de salida.

Tabla 10

Estructura de segundo modelo entrenado.

Capa	Salida	Parámetros
conv2d_1 (Conv2D)	(None, 224, 224, 32)	896
max_pooling2d_1(MaxPooling2)	(None, 112, 112, 32)	0
conv2d_2 (Conv2D)	(None, 112, 112, 64)	8256
max_pooling2d_2(MaxPooling2)	(None, 56, 56, 64)	0
conv2d_3 (Conv2D)	(None, 56, 56, 128)	32896
max_pooling2d_3(MaxPooling2)	(None, 28, 28, 128)	0
flatten_1 (Flatten)	(None, 200704)	0
dense_1 (Dense)	(None, 256)	25690368
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 5)	1285

Fuente: Elaboración propia.

Otra con 4 capas de convolución y 4 capas de maxpooling, seguido de una capa totalmente conectada y una capa de salida.

Tabla 11

Estructura de tercer modelo entrenado.

Capa	Salida	Parámetros
conv2d_1 (Conv2D)	(None, 224, 224, 32)	896
max_pooling2d_1(MaxPooling2)	(None, 112, 112, 32)	0
conv2d_2 (Conv2D)	(None, 112, 112, 64)	8256
max_pooling2d_2(MaxPooling2)	(None, 56, 56, 64)	0
conv2d_3 (Conv2D)	(None, 56, 56, 128)	32896
max_pooling2d_3(MaxPooling2)	(None, 28, 28, 128)	0
conv2d_4 (Conv2D)	(None, 28, 28, 256)	131328
max_pooling2d_4(MaxPooling2)	(None, 14, 14, 256)	0
flatten_1 (Flatten)	(None, 200704)	0
dense_1 (Dense)	(None, 256)	12845312
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 5)	1285

Fuente: Elaboración propia.

El modelo VGG16 que está preparada para recibir imágenes de 224x224x3 (ancho * alto * RGB), tiene 13 capas de convolución, 5 de ellas con maxpooling, tres capas de redes profundas, y por último una capa de softmax para detectar las 5 clases.

Tabla 12

Estructura de modelo VGG16.

Capa	Salida	Parámetros
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080

block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
dense_1 (Dense)	(None, 5)	5005

Fuente: Elaboración propia.

Entrenamiento de modelo con 2 capas de convolución:

En la etapa de entrenamiento se logró observar que los resultados de exactitud oscilan entre 94% y 98%, y los de perdida están entre los datos obtenidos se muestran a continuación.

Tabla 13

Resultados de exactitud y perdida de etapa de entrenamiento del primer modelo de prueba.

	Exactitud		**Perdida**	
Época	**Entrenamiento**	**Validación**	**Entrenamiento**	**Validación**
1	0.9492	0.9881	0.3156	0.0692
2	0.9954	0.9881	0.0152	0.0280
3	0.9996	0.9944	0.0018	0.0194
4	0.9976	0.9821	0.0091	0.1311
5	0.9901	0.9881	0.0345	0.0779
6	0.9978	0.9884	0.0065	0.1047

7	0.9999	0.9884	3.3720e-04	0.0504
8	1	0.9881	1.9998e-04	0.0317
9	0.9997	0.9881	5.9042e-04	0.1584
10	0.9911	0.9944	0.0389	0.0699
11	0.9973	0.9940	0.0115	0.0962
12	0.9975	0.9940	0.0116	0.0963
13	0.9995	0.9940	0.0025	0.0962
14	0.9972	0.9940	0.0105	0.0965
15	0.9961	0.9881	0.0230	0.1379
16	0.9971	0.9881	0.0143	0.1030
17	0.9985	0.9940	0.0056	0.0962
18	1	0.9940	4.1619e-05	0.0962
19	0.9956	0.9884	0.0307	0.1868
20	1	0.9881	2.6667e-05	0.1893

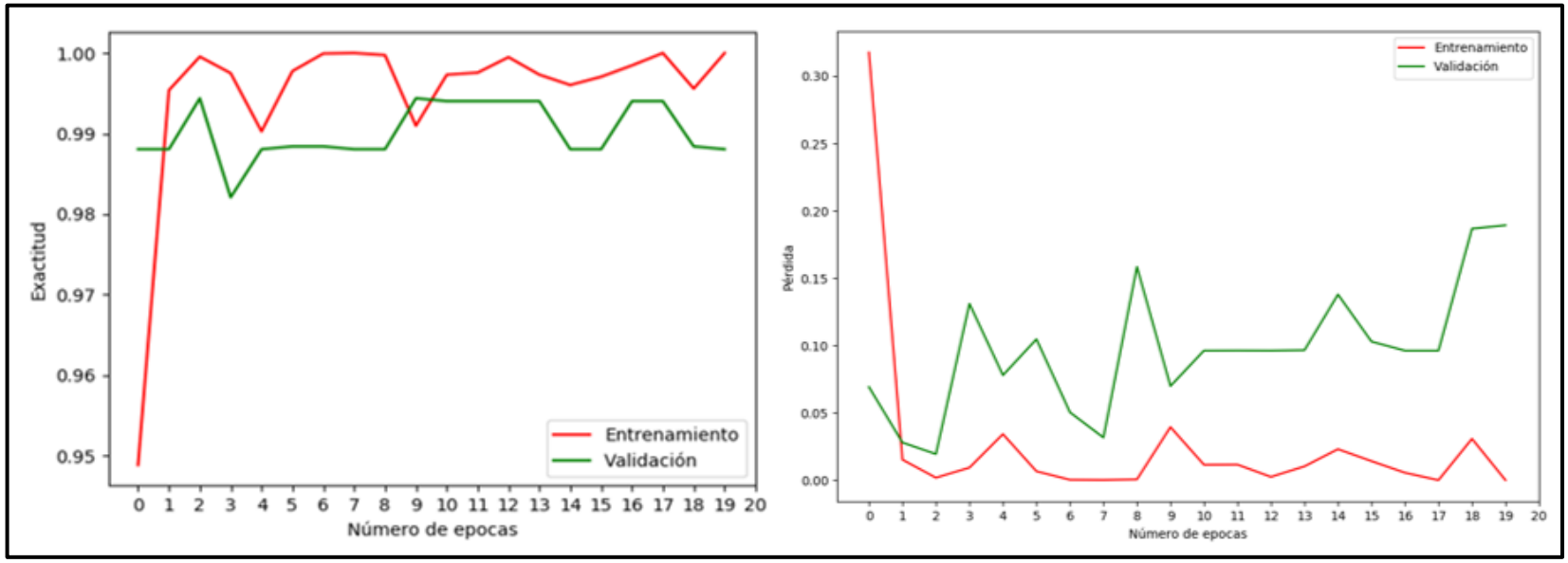

Figura 32. Porcentaje de exactitud y pérdida de entrenamiento y de validación en el primer modelo de prueba.

Entrenamiento de modelo con 3 capas de convolución:

En la etapa de entrenamiento se logró observar que los resultados oscilan entre 96% y 99%, los datos obtenidos se muestran a continuación.

Tabla 14

Resultados de exactitud y perdida de etapa de entrenamiento del segundo modelo de prueba.

	Exactitud		Perdida	
Época	**Entrenamiento**	**Validación**	**Entrenamiento**	**Validación**
1	0.9630	0.9824	0.1366	0.0770
2	0.9929	0.9765	0.0208	0.1126
3	0.9943	0.9824	0.0216	0.0640
4	0.9971	0.9884	0.0099	0.1012
5	0.9992	0.9940	0.0021	0.0881
6	0.9900	0.9824	0.0495	0.1949
7	0.9958	0.9940	0.0209	0.0468
8	0.9979	0.9884	0.0074	0.0451
9	0.9994	1	0.0013	2.6462e-04
10	0.9950	0.9828	0.0286	0.2277
11	0.9942	0.9840	0.0343	0.0967
12	0.9988	0.9888	0.0038	0.1720
13	0.9987	0.9884	0.0063	0.1868
14	0.9959	0.9881	0.0244	0.1595
15	0.9980	0.9881	0.0111	0.1037
16	0.9970	0.9881	0.0226	0.1841
17	0.9919	0.9765	0.0795	0.2070
18	0.9984	0.9940	0.0150	0.0962
19	0.9989	0.9881	0.0077	0.1730
20	0.9976	0.9881	0.0165	0.1924

Fuente: Elaboración propia.

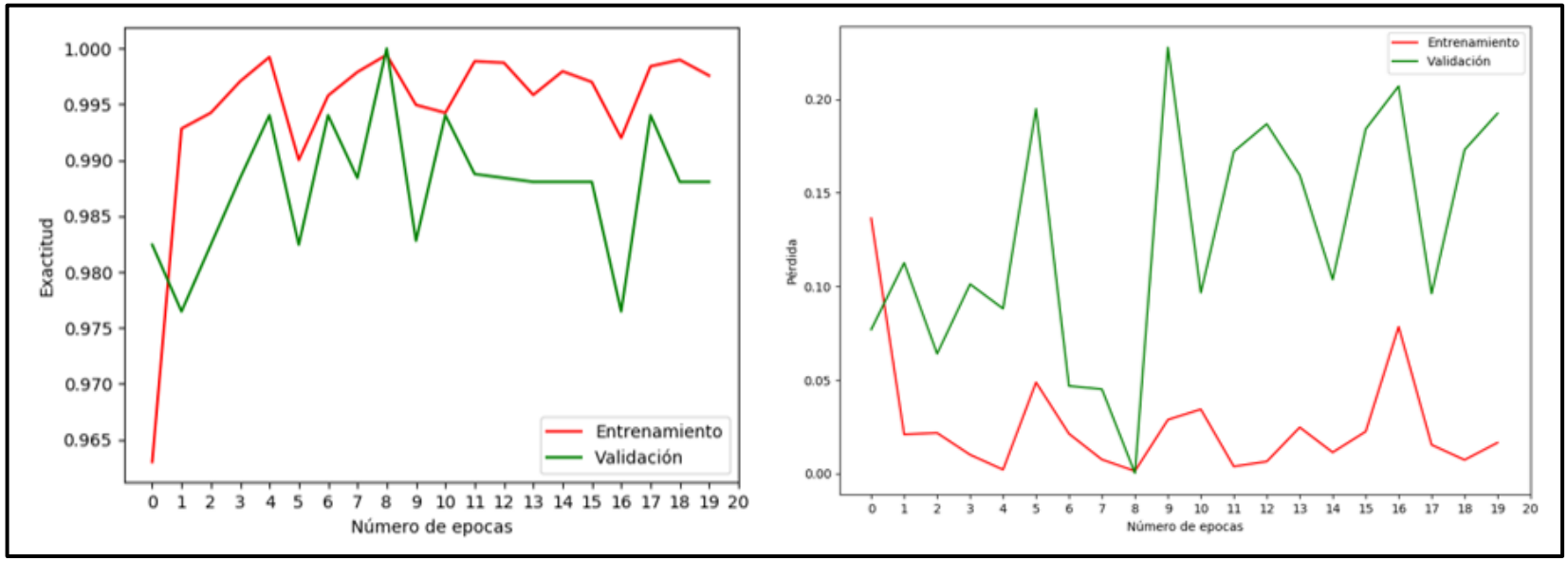

Figura 33. Porcentaje de exactitud y pérdida de entrenamiento y de validación en segundo modelo de prueba.

Entrenamiento de modelo con 4 capas convolución:

En la etapa de entrenamiento se logró observar que los resultados oscilan entre 96% y 99%, los datos obtenidos se muestran a continuación.

Tabla 15

Resultados de exactitud y perdida de etapa de entrenamiento del tercer modelo de prueba.

	Exactitud		**Perdida**	
Época	Entrenamiento	Validación	Entrenamiento	Validación
1	0.9610	0.9944	0.1153	0.0914
2	0.9903	0.9944	0.0388	0.0922
3	0.9986	0.9944	0.0046	0.0914
4	0.9807	0.9821	0.1482	0.1087
5	0.9964	0.9884	0.0172	0.1525
6	0.9996	0.9884	0.0011	0.1135
7	0.9964	0.9940	0.0180	0.0540
8	0.9990	0.9940	0.0048	0.0967
9	0.9839	0.9940	0.1565	0.0962
10	0.9956	0.9888	0.0371	0.1252
11	0.9938	0.9881	0.0557	0.1173
12	0.9958	0.9944	0.0411	0.0906
13	0.9981	0.9940	0.0177	0.0962
14	0.9988	0.9940	0.0115	0.0962
15	0.9989	0.9881	0.0091	0.1928
16	0.9919	0.9824	0.0953	0.2697
17	0.9937	1	0.0752	1.1921e-07
18	0.9864	1	0.1823	1.1921e-07
19	0.9954	1	0.0647	1.1921e-07
20	0.9981	0.9824	0.0251	0.2830

Fuente: Elaboración propia.

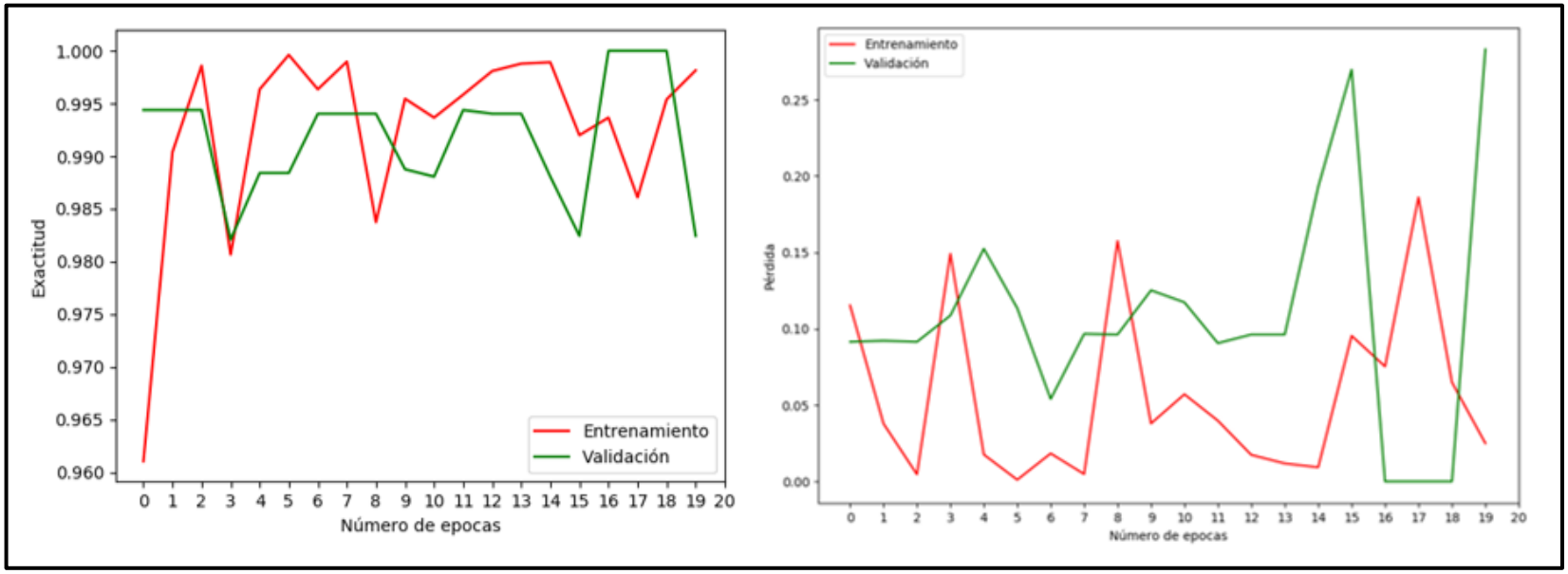

Figura 34. Porcentaje de exactitud y pérdida de entrenamiento y de validación en tercer modelo de prueba.

Entrenamiento de modelo VGG16:

En la etapa de entrenamiento del modelo VGG16 se logró observar que los resultados oscilan entre 65% y 71%, los datos obtenidos se muestran a continuación.

Tabla 16

Resultados de exactitud de etapa de entrenamiento del cuarto modelo de prueba.

	Exactitud		Perdida	
Época	**Entrenamiento**	**Validación**	**Entrenamiento**	**Validación**
1	0.5383	0.7110	1.6018	1.5933
2	0.6442	0.6647	1.5872	1.5777
3	0.6309	0.6131	1.5729	1.5622
4	0.6413	0.6577	1.5590	1.5477
5	0.6518	0.6703	1.5452	1.5331
6	0.6537	0.6397	1.5319	1.5189
7	0.6529	0.6482	1.5192	1.5046
8	0.6573	0.6520	1.5064	1.4916
9	0.6579	0.6527	1.4940	1.4779
10	0.6541	0.6541	1.4822	1.4649
11	0.6538	0.6531	1.4716	1.4524
12	0.6601	0.6636	1.4592	1.4404
13	0.6650	0.6577	1.4479	1.4281
14	0.6598	0.6591	1.4364	1.4163
15	0.6644	0.6657	1.4259	1.4045
16	0.6660	0.6664	1.4149	1.3937
17	0.6658	0.6594	1.4055	1.3829
18	0.6699	0.6622	1.3949	1.3732
19	0.6651	0.6629	1.3861	1.3616
20	0.6771	0.6699	1.3762	1.3514

Fuente: Elaboración propia.

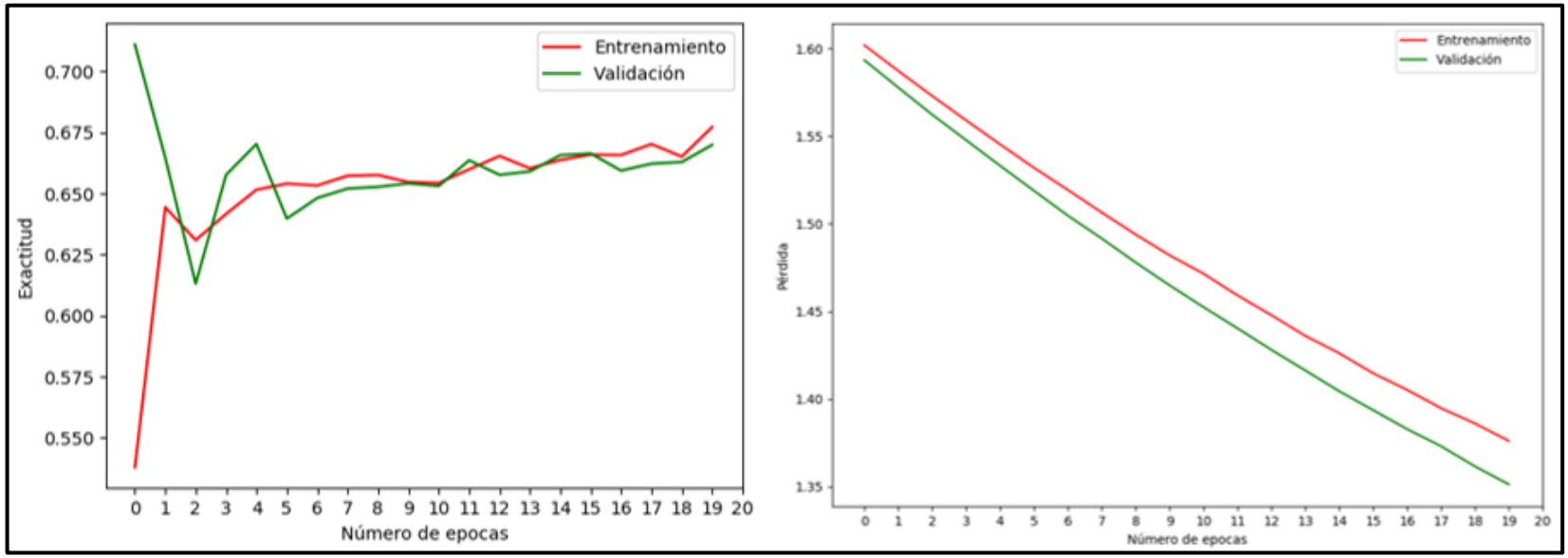

Figura 35. Porcentaje de exactitud y pérdida de entrenamiento y de validación en modelo VGG16.

Tabla 17

Valores de Precisión - recall y exactitud del primer modelo de prueba

	Precisión	Recall
Capi	1	0.94
Chimuelo	0.55	1
Mauro	1	0.79
Tomas	0.92	0.33
Yui	0.89	0.97
Exactitud		0.81

Fuente: Elaboración propia.

Tabla 18

Valores de Precisión - recall y exactitud del segundo modelo de prueba

	Precisión	Recall
Capi	0.97	0.97
Chimuelo	0.94	1
Mauro	1	1
Tomas	1	0.97
Yui	1	0.97
Exactitud		0.98

Fuente: Elaboración propia.

Tabla 19

Valores de Precisión - recall y exactitud del tercer modelo de prueba

	Precisión	Recall
Capi	1	0.91
Chimuelo	0.87	1
Mauro	1	1
Tomas	1	0.97
Yui	0.97	0.94
Exactitud		0.96

Fuente: Elaboración propia.

Tabla 20

Valores de Precisión - recall y exactitud del modelo de VGG16

	Precisión	Recall
Capi	0	0
Chimuelo	0.06	0.21
Mauro	0.08	0.06
Tomas	0	0
Yui	1	0.39
Exactitud		0.13

Fuente: Elaboración propia.

Construir el modelo de la red neuronal convolucional.

Luego que se realizó las pruebas se eligió trabajar con el segundo modelo debido a los resultados obtenidos.

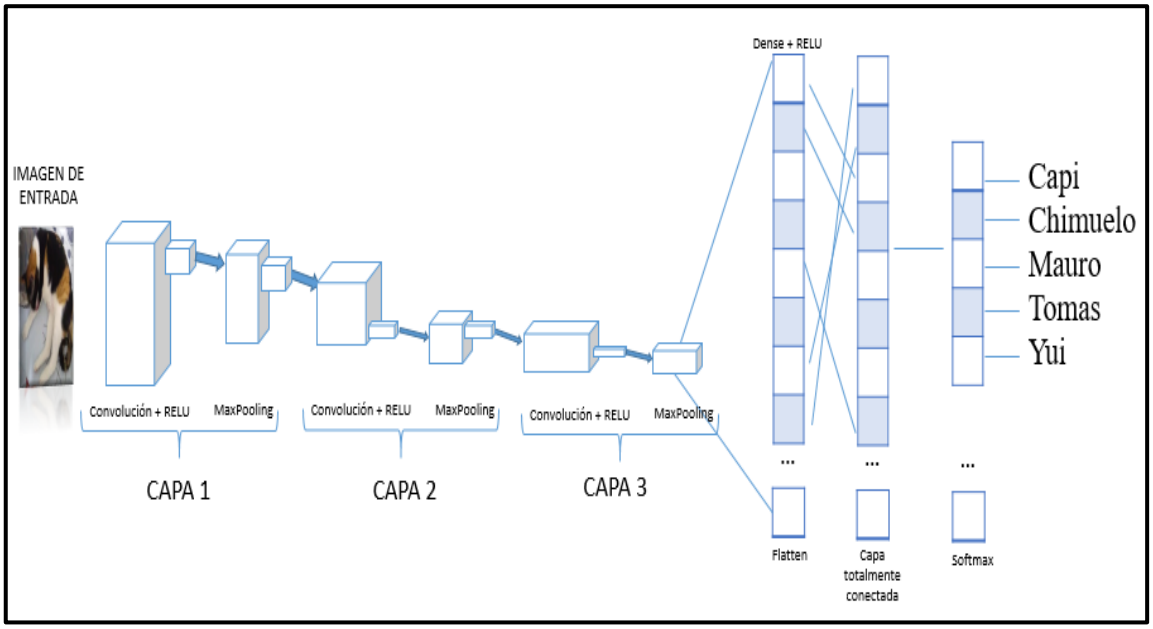

Figura 36. Red neuronal convolucional de 3 capas.

Antes de que se comience con la etapa de entrenamiento se realizó una etapa de preparación de las imágenes para hacer que el entrenamiento sea más eficiente.

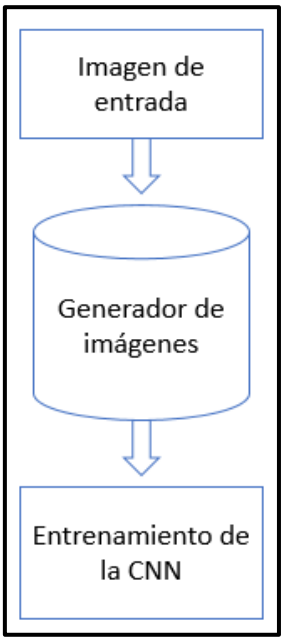

Figura 37. Etapa de pre procesamiento de las imágenes antes de pasar a la etapa de entrenamiento.

Lo que se busca al implementar este segmento de código es que las imágenes de entrada tengan un pre procesamiento antes de pasar a la etapa de entrenamiento, para ayudar a que nuestro modelo aprenda a que no siempre las imágenes van a estar en una misma posición.

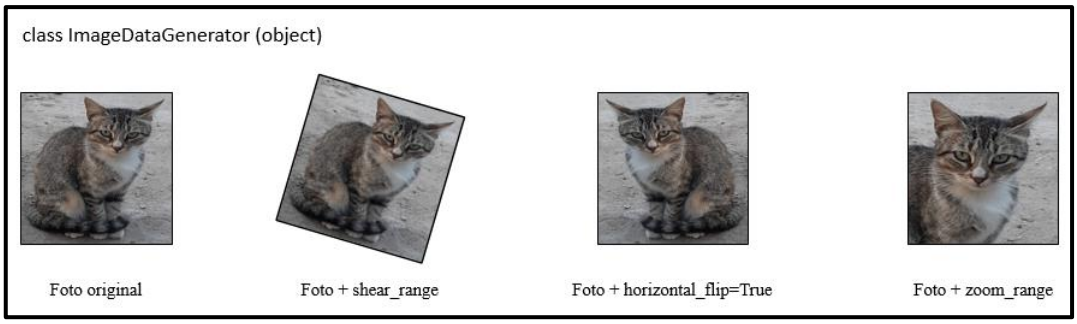

Figura 38. Preparación de imágenes.

Luego que se hizo el procesamiento de la imagen, la imagen pasa por las capas para que se pueda obtener sus patrones.

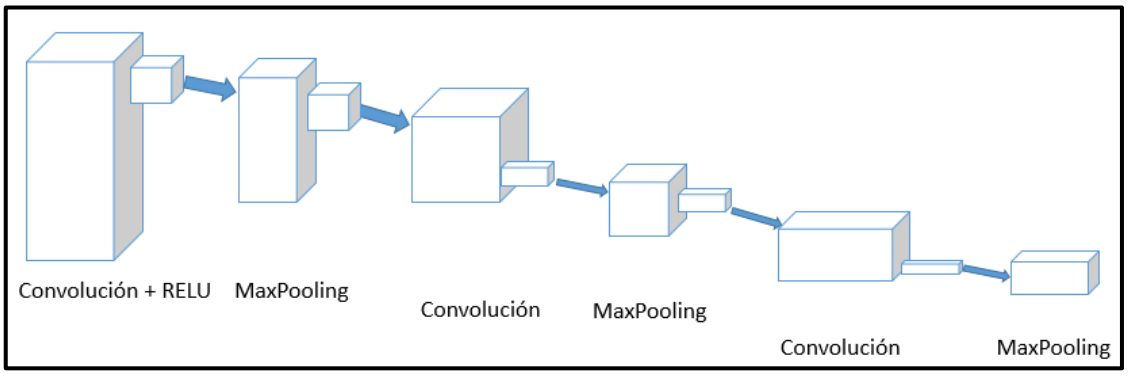

Figura 39. Modelo de 3 capas de convolución y maxpooling.

Se creó la primera capa de convolución se colocó padding *"same"* para que coloque 0 al contorno de la imagen, se usó la función de activación *"relu"* para darle no linealidad, para que cuando los valores de entrada sean negativos los convierta en 0. En esta capa cada uno de los filtros buscan características, por ejemplo, líneas, texturas, etc.

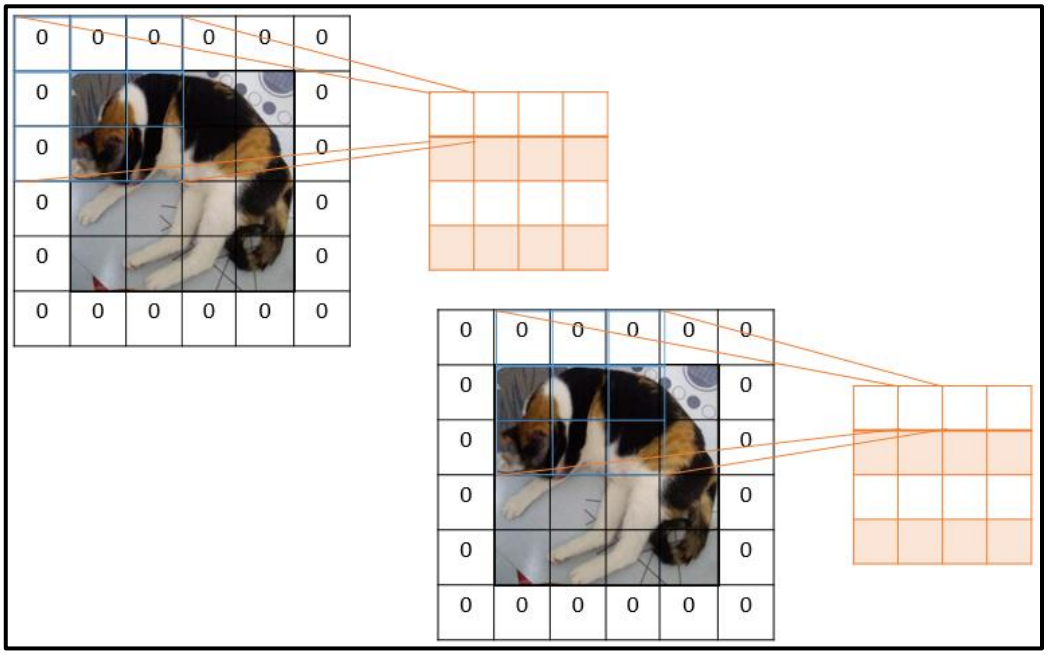

Figura 40. Proceso de convolución.

Se utilizó maxpooling para reducir la dimensión de la imagen, recorriendo con un filtro de 2x2 en toda la imagen y sacando el mayor valor.

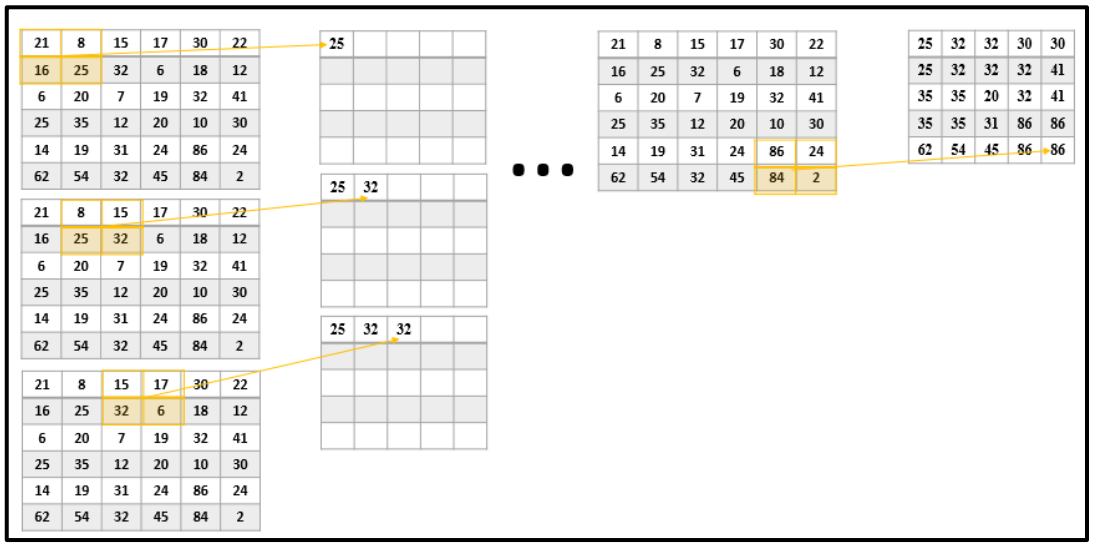

Figura 41. Proceso de maxpooling.

Los patrones obtenidos en las capas anteriores luego se conectan en una capa totalmente conectada para que se tenga patrones más complejos.

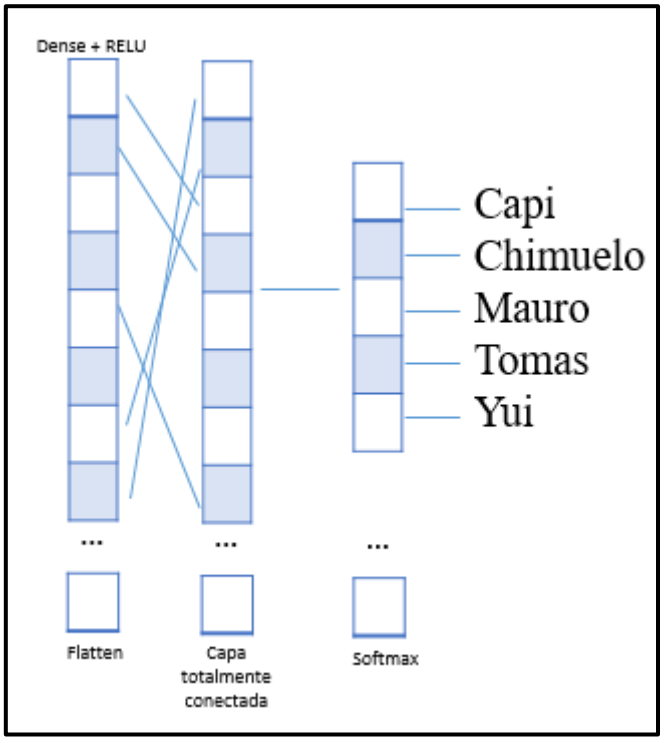

Figura 42. Capa aplanada, capa totalmente conectada y capa de salida con función de activación softmax.

Se utilizó la función de activación softmax porque se trabajó con múltiples clases, su función es dar un porcentaje de probabilidades de que una imagen pertenece a una clase en específico. La fórmula de la función softmax es:

$$\frac{e^{Y_j}}{\sum_{k=1}^{K}(e^{Y_k})}$$

Donde Y es la salida de las capas ocultas y K es el número de clases en el modelo. Los valores 0.4, 0.5, 0.45, 0.1 se reciben de las capas ocultas y luego pasan por la función de activación softmax, donde la exponencial del primer valor es dividida por la sumatoria de todas las exponenciales.

0.4 -> softmax -> $\frac{e^{0.4}}{e^{0.4}+e^{0.5}+e^{0.45}+e^{0.1}} = 0.2565905122 = 25.66\%$

0.5 -> softmax -> $\frac{e^{0.5}}{e^{0.4}+e^{0.5}+e^{0.45}+e^{0.1}} = 0.283576372 = 28.36\%$

0.45 -> softmax -> $\frac{e^{0.45}}{e^{0.4}+e^{0.5}+e^{0.45}+e^{0.1}} = 0.2697461891 = 26.97\%$

0.1 -> softmax -> $\frac{e^{0.1}}{e^{0.4}+e^{0.5}+e^{0.45}+e^{0.1}} = 0.1900869267 = 19.01\%$

Evaluar resultados

Luego que se terminó con la etapa de entrenamiento, se procedió a probar el modelo entrenado, para ello se implementó un código de predicción. Con el modelo ya cargado enviamos la foto que se deseó predecir y por resultado se obtuvo un arreglo entre ceros y unos donde 1 marcaba la posición de la clase predicha.

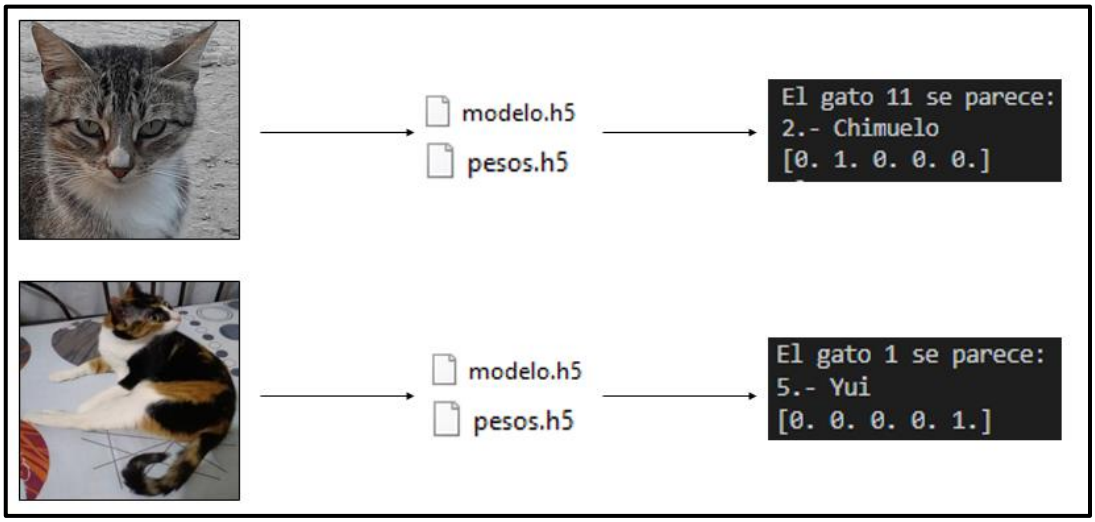

Figura 43. Modelo de etapa de predicción.

Luego que se realizó la etapa de predicción se armó la matriz de confusión para evaluar que tan bueno es nuestro modelo entrenado.

Tabla 21

Matriz de confusión de entrenamiento de modelo de 3 capas

		Verdaderos				
		Capi	Chimuelo	Mauro	Tomas	Yui
Predicciones	Capi	32	1	0	0	0
	Chimuelo	0	33	0	0	0
	Mauro	0	0	33	0	0
	Tomas	1	0	0	32	0
	Yui	0	1	0	0	32

Fuente: Elaboración propia.

Luego que se obtuvo los resultados y se plasmaron en la matriz de confusión se procedió a evaluar el nivel de precisión global y por clase, además de la sensibilidad y exactitud.

Donde:

TP = Verdaderos positivos

TN = Verdaderos negativos.

FP = Falsos positivos.

FN = Falsos negativos.

Exactitud:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Precisión global:

$$\frac{suma\ de\ diagonal\ principal}{TP + TN + FP + FN}$$

$$\frac{32 + 33 + 33 + 32 + 32}{\begin{matrix}32 + 1 + 0 + 0 + 0 + 0 + 33 + 0 + 0 + 0 + 0 + 0 + 33 + 0 + 0 + 1 + 0 + 0 + 32 + 0 \\ +0 + 1 + 0 + 0 + 32\end{matrix}}$$

$$\frac{162}{165} = 0{,}98$$

Precisión para la clase capi:

$$\frac{TP}{TP + FP}$$

$$\frac{32}{32 + 1}$$

$$\frac{32}{33} = 0{,}97$$

Precisión para la clase chimuelo:

$$\frac{TP}{TP + FP}$$

$$\frac{33}{33 + 2}$$

$$\frac{33}{35} = 0{,}94$$

Precisión para la clase mauro:

$$\frac{TP}{TP + FP}$$

$$\frac{33}{32 + 1}$$

$$\frac{33}{33} = 1$$

Precisión para la clase tomas:

$$\frac{TP}{TP + FP}$$

$$\frac{32}{32}$$

$$\frac{32}{32} = 1$$

Precisión para la clase yui:

$$\frac{TP}{TP + FP}$$

$$\frac{32}{32 + 1}$$

$$\frac{32}{32} = 1$$

Recall para la clase capi:

$$\frac{TP}{TP + FN}$$

$$\frac{32}{32+1}$$

$$\frac{32}{33} = 0{,}97$$

Recall para la clase chimuelo:

$$\frac{TP}{TP+FP}$$

$$\frac{33}{32+1}$$

$$\frac{33}{33} = 1$$

Recall para la clase mauro:

$$\frac{TP}{TP+FP}$$

$$\frac{33}{32+1}$$

$$\frac{33}{33} = 1$$

Recall para la clase tomas:

$$\frac{TP}{TP+FP}$$

$$\frac{32}{32+1}$$

$$\frac{32}{33} = 0{,}97$$

Recall para la clase yui:

$$\frac{TP}{TP+FP}$$

$$\frac{32}{32+1}$$

$$\frac{32}{33} = 0{,}97$$

IV. CONCLUSIONES Y RECOMENDACIONES

Conclusiones:

Basado en los resultados obtenidos de la implementación de este trabajo, se llega a la conclusión de:

En la adquisición de las imágenes la distancia prudente para la toma de las fotos es de 2 metros para que no ahuyentar al gato.

En la etapa armado del dataset la cantidad de imágenes por cada clase deben ser la misma, porque si una clase sobresale de otra está va atender mucho a confundirse entre las otras clases al momento de realizar el entrenamiento.

En la etapa de implementación se llega a la conclusión de que la estructura de la red neuronal convolucional puede variar según la cantidad de imágenes que se tenga, además de que usar las librerías de tensorflow y keras hace que esta implementación sea mucho más fácil y sencilla de realizar.

En el objetivo general se concluye que las redes neuronales convolucionales son efectivas llegando a obtener un 98% de exactitud en este trabajo, teniendo en cuenta que las fotos no tenían la variedad como para que las capas puedan obtener diferentes patrones.

Recomendaciones:

Para trabajos futuros se recomienda utilizar las siguientes versiones:

Python 3.6.8, Tensorflow 1.8.0, Keras 2.2.4, numpy 1.16.3, cuda 9.0.

Pero para si desea utilizar transferencia de aprendizaje se recomienda instalar la versión de keras 2.1.6 porque esta versión fue la que permitió entrenar el modelo, predecir las imágenes y también poder obtener su estructura del modelo.

Se recomienda al momento de realizar la toma de fotos, que las fotos de todos los gatos se hagan en las mismas circunstancias, con la misma iluminación, con cámaras que tengan mejor resolución, buscando diferentes ángulos, tratando de que el gato sobresalga del fondo, por ejemplo, si es un gato negro evitar realizar la toma en una zona con sombra.

Si se desea trabajar con detección usando librerías de opencv, se recomienda que estas librerías sean instaladas en un ambiente de trabajo diferente a donde se está trabajando la clasificación.

REFERENCIAS

Quintero, C., Merchán, F., Cornejo, A., & Galán, J. S. (2018). Uso de Redes Neuronales Convolucionales para el Reconocimiento Automático de Imágenes de Macroinvertebrados para el Biomonitoreo Participativo. *KnE Engineering*, *3*(1), 585. https://doi.org/10.18502/keg.v3i1.1462

Araujo, A., Pérez, J., & Rodriguez, W. (2018). Aplicación de una Red Neuronal Convolucional para el Reconocimiento de Personas a Través de la Voz. *Sexta Conferencia Nacional de Computación, Informática y Sistemas*, 77–81.

Jiménez Moreno, R., Martínez Baquero, J. E., & Rodríguez Umaña, L. A. (2018). Automatic fish classification system. *Revista Visión Electrónica*, *12*(2), 258–264. https://doi.org/10.14483/22484728.14265

Myat Mon Kyaw | San San Nwe | Myint Myint Yee. (2019). Pest Classification and Pesticide Recommendation System. *International Journal of Trend in Scientific Research and Development (IJTSRD) International Journal of Trend in Scientific Research and Development*, *3*(5), 2187–2191. https://doi.org/https://doi.org/10.31142/ijtsrd27899

Picazo Montoya, Ó. (2018). *Redes Neuronales Convolucionales Profundas para el reconocimiento de emociones en imágenes*. 45. Retrieved from http://oa.upm.es/51441/%0Ahttp://oa.upm.es/51441/1/TFM_OSCAR_PICAZO_MONTOYA.pdf

Ascarza Mendoza, F. J. (2018). *Segmentación Automática de Textos, mediante Redes Neuronales Convolucionales en Imágenes Documentos Históricos*. 67. Retrieved from http://tesis.pucp.edu.pe/repositorio/handle/123456789/13050

Martínez Llamas, J. (n.d.). *Escuela Tecnnica Supeirior de Ingennieria de Sitemas Informaticos.* Retrieved from http://oa.upm.es/53050/1/TFG_JAVIER_MARTINEZ_LLAMAS.pdf

Durán Suárez, J., Del, A., Torres, R., & Suárez, D. (2017). *Redes Neuronales Convolucionales en R Reconocimiento de caracteres escritos a mano Redes Neuronales Convolucionales en R Reconocimiento de caracteres escritos a mano Redes Neuronales Convolucionales en R*. 78. Retrieved from http://bibing.us.es/proyectos/abreproy/91338/fichero/TFG+Jaime+Durán+Suárez.pdf

Vizcaya, R., Martin, J., Albino, F., & Lazcano-Salas, S. (2017). *Desempeño de una red neuronal convolucional para clasificación de señales de tránsito*. *5*, 795–802. Retrieved from https://www.researchgate.net/publication/323456954

Krishna, Ds., & Jena, B. (2019). A Trained CNN Based Resolution Enhancement of Digital Images. *International Journal for Modern Trends in Science and*

Technology, *05*(7), 9–16. Retrieved from http://www.ijmtst.com/vol5issue07.html

Kumar, S., Tiwari, S., & Singh, S. K. (2016). Face recognition for cattle. *Proceedings of 2015 3rd International Conference on Image Information Processing, ICIIP 2015*, 65–72. https://doi.org/10.1109/ICIIP.2015.7414742

Mukai, N., Zhang, Y., & Chang, Y. (2018). Pet face detection. *Proceedings - 2018 NICOGRAPH International, NICOINT 2018*, 52–57. https://doi.org/10.1109/NICOINT.2018.00018

Loos, A., & Kalyanasundaram, T. A. M. (2015). *FACE RECOGNITION FOR GREAT APES: IDENTIFICATION OF PRIMATES IN VIDEOS Alexander Loos and Talat Anand Mohan Kalyanasundaram Fraunhofer Institute for Digital Media Technology IDMT Audio-Visual Systems*. 1548–1552.

ANEXOS

Resolución de Ampliación de vigencia de tesis

UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO
RESOLUCIÓN N°0570-2022/FIAU-USS

Pimentel, 19 de septiembre de 2022

VISTO:
El Acta de reunión N°0509-2022 del Comité de investigación de la Escuela profesional de INGENIERÍA DE SISTEMAS remitida mediante Oficio 0195-2022/FIAU-IS-USS de fecha 5 de septiembre de 2022, y;

CONSIDERANDO:

Que, de conformidad con la Ley Universitaria N° 30220 en su artículo 48° que a letra dice: "La investigación constituye una función esencial y obligatoria de la universidad, que la fomenta y realiza, respondiendo a través de la producción de conocimiento y desarrollo de tecnologías a las necesidades de la sociedad, con especial énfasis en la realidad nacional. Los docentes, estudiantes y graduados participan en la actividad investigadora en su propia institución o en redes de investigación nacional o internacional, creadas por las instituciones universitarias públicas o privadas.";

Que, de conformidad con el Reglamento de grados y títulos en su artículo 21° señala: "Los temas de trabajo de investigación, trabajo académico y *tesis* son *aprobados por el Comité de Investigación* y derivados a la facultad o Escuela de Posgrado, según corresponda, para la emisión de la resolución respectiva. El *periodo de vigencia de los mismos será de dos años*, a partir de su aprobación. En caso un tema perdiera vigencia, el Comité de Investigación evaluará la ampliación de la misma.

Que, de conformidad con el Reglamento de grados y títulos en su artículo 24° señala: La tesis es un estudio que debe denotar rigurosidad metodológica, originalidad, relevancia social, utilidad teórica y/o práctica en el ámbito de la escuela profesional. Para el grado de doctor se requiere una tesis de máxima rigurosidad académica y de carácter original. Es individual para la obtención de un grado; *es individual o en pares para obtener un título profesional*. Asimismo, en su artículo 25° señala: "El tema debe responder a alguna de las líneas de investigación institucionales de la USS S.A.C.".

Facultad de Ingeniería, Arquitectura y Urbanismo
UNIVERSIDAD SEÑOR DE SIPÁN SAC.

Que, mediante documentos de vistos, el Comité de investigación de la referida Escuela profesional acordó aprobar la ampliación de la vigencia de las tesis que se detallan en el Acta de reunión N° 0509 - 2022, de la línea de investigación de INFRAESTRUCTURA, TECNOLOGÍA Y AMBIENTE, a cargo de los estudiantes y /o egresados del Programa de estudios INGENIERÍA DE SISTEMAS, hasta la fecha que indica la presente resolución.

Estando a lo expuesto, y en uso de las atribuciones conferidas y de conformidad con las normas y reglamentos vigentes;

SE RESUELVE:

ARTÍCULO ÚNICO: AMPLIAR VIGENCIA, de la Tesis a cargo de los estudiantes y /o egresados del Programa de estudios de **INGENIERÍA DE SISTEMAS** que se detallan en el anexo de la presente Resolución.

UNIVERSIDAD SEÑOR DE SIPÁN

FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO
RESOLUCIÓN N°0570-2022/FIAU-USS

Pimentel, 19 de septiembre de 2022

ANEXO

APELLIDOS	TESIS	AMPLIACION HASTA
ALARCON GUEVARA IVON VANESSA RAMOS MENDOZA NADALI GLORIA	EVALUACIÓN DE LA CALIDAD EN USO DE PLATAFORMAS DE COMERCIO ELECTRÓNICO BASADAS EN SOFTWARE LIBRE CON MAYOR USO POR EMPRESAS PERUANAS, UTILIZANDO EL ESTANDAR ISO/IEC 25000	31 DE DICIEMBRE DEL 2022
BARTENS AMARO VALERY FERNANDO	ANÁLISIS COMPARATIVO DE TÉCNICAS DE INTEGRACIÓN DE PLATAFORMAS PARA LA OBTENCION DE INFORMACION INTEGRADA	31 DE DICIEMBRE DEL 2022
HERNANDEZ NERIA MARCO ANTONIO ROSAS	IDENTIFICACIÓN AUTOMÁTICA DE GATOS MEDIANTE RECONOCIMIENTO DE IMÁGENES USANDO REDES NEURONALES CONVOLUCIONALES	31 DE DICIEMBRE DEL 2022
ANTON CHICLAYO RAFAEL JHAMYR	IDENTIFICACIÓN AUTOMÁTICA DE PERSONAS MEDIANTE EL PROCESAMIENTO DE IMÁGENES DIGITALES DE LAS LÍNEAS PALMARIAS	31 DE DICIEMBRE DEL 2022
SIESQUEN SANDOVAL PABLO	DESARROLLO DE UN MODELO DE GESTIÓN DE INCIDENTES DE TI BASADO EN ESTÁNDARES DE BUENAS PRÁCTICAS PARA MEJORAR EL SERVICIO DE TI EN LAS DIRECCIONES REGIONALES DE SALUD DEL PERÚ	31 DE JULIO DEL 2023
RABINES PANDURO JHANET	DISEÑO DE UN MODELO DE CALIDAD DE SERVICIOS PARA LA MESA DE AYUDA DE LA MUNICIPALIDAD DISTRITAL DE SAN MARTIN	31 DE DICIEMBRE DEL 2022
TANTAJULCA ROJAS NERLITA MARDELI CIEZA RIOS ELMER	DESARROLLO DE UN PROCESO DE PRUEBAS BASADA EN ESTÁNDARES PARA MEJORAR LA EFICIENCIA DE LA VERIFICACIÓN Y VALIDACIÓN DEL PRODUCTO EN MICRO EMPRESAS PERUANAS QUE DESARROLLAN SOFTWARE	31 DE JULIO DEL 2023
FERNANDEZ IRIGOIN JOSE ARMANDO NUÑEZ CAYOTOPA JOSE JILMER	EVALUACIÓN DE LA USABILIDAD DEL SISTEMA DE INFORMACIÓN DE APOYO A LA GESTIÓN DE LA INSTITUCIÓN EDUCATIVA DEL MINISTERIO DE EDUCACIÓN PERUANO MEDIANTE LA NORMA ISO/IEC 25010	31 DE JULIO DEL 2023
CASTAÑEDA ALARCON FRANKLIN EDWARD PUELLES RUIZ RONALD ROBESPIERRE	DESARROLLO DE UN PROCESO DE PRUEBAS BASADA EN ESTÁNDARES PARA MEJORAR LA EFICIENCIA DE LA VERIFICACIÓN Y VALIDACIÓN DEL PRODUCTO EN MICRO EMPRESAS PERUANAS QUE DESARROLLAN SOFTWARE	31 DE JULIO DEL 2023
LARA PERLECHE LOURDES PATRICIA	EVALUACION DE LA USABILIDAD EN ENTORNOS VIRTUALES DE APRENDIZAJE MEDIANTE LA NORMA ISO/IEC 25023-2016. CASO DE ESTUDIO UNIVERSIDAD SEÑOR DE SIPAN.	31 DE JULIO DEL 2023
FLORES TELLO JAIME NICOLAS	DETECCIÓN AUTOMÁTICA DE LA ENFERMEDAD LASIODIPLODIA THEOBROMAE DEL PALTO UTILIZANDO IMÁGENES DIGITALES CON REDES NEURONALES CONVOLUCIONALES	31 DE DICIEMBRE DEL 2022
CHAVEZ MANAYALLE JHON SEBASTY JHAIR GUTIERREZ BALCAZAR GRABIELA YUDITH	DESARROLLO DE UNA METODOLOGÍA DE GESTIÓN DE INCIDENCIAS BASADO EN MARCOS DE REFERENCIA PARA MITIGAR LA INTERRUPCIÓN DE SERVICIOS DE TI EN PYMES DEL SECTOR DE CONSTRUCCIÓN EN EL PERÚ: CASO DE ESTUDIO KIBE CONSTRUCCIONES S.A.C	31 DE JULIO DEL 2023
RUIZ SANTA CRUZ YOVANNY FLORIBEL MARIA ROJANA SANCHEZ TORRES	DESARROLLO DE UN MÉTODO DE CLASIFICACIÓN AUTOMÁTICA DE LOS DEFECTOS EXTERNOS DE SOLANUM PHUREJA PARA CUMPLIR CON LAS EXIGENCIAS DE CALIDAD DEL MERCADO PERUANO UTILIZANDO PROCESAMIENTO DE IMÁGENES DIGITALES Y APRENDIZAJE DE MÁQUINA	31 DE JULIO DEL 2023

DR. VICTOR ALEXCI TUESTA MONTEZA
DECANO (E) FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO
UNIVERSIDAD SEÑOR DE SIPÁN SAC.
CHICLAYO

DR. HALYN ALVAREZ VÁSQUEZ
SECRETARIO ACADÉMICO | FACULTAD DE INGENIERÍA, ARQUITECTURA Y URBANISMO
UNIVERSIDAD SEÑOR DE SIPÁN SAC.
CHICLAYO

REGÍSTRESE, COMUNÍQUESE Y ARCHÍVESE

Implementación de la red neuronal convolucional:

Para implementar la cnn se trabajó en la interfaz de usuario anaconda navigator, se eligió anaconda navigator porque permitió crear diferentes ambientes de trabajo donde puedes instalar las bibliotecas necesarias para un trabajo determinado, además que fue muy fácil de utilizar y lo que se necesitó se pudo instalar por terminal de manera muy sencilla. El entorno de trabajo que se utilizó fue visual studio code y se trabajó con el lenguaje de programación python. Se utilizó las librerías de tensorflow y keras porque permitieron implementar una red neuronal convolucional de manera muy sencilla. Las características del ordenador que se utilizó son: Procesador AMD A8-7600, 2 memorias RAM de 8GB HyperX DDR3 y una tarjeta gráfica nvidia GeForce GTX 1050 TI de 4gb. Sistema operativo Windows 10

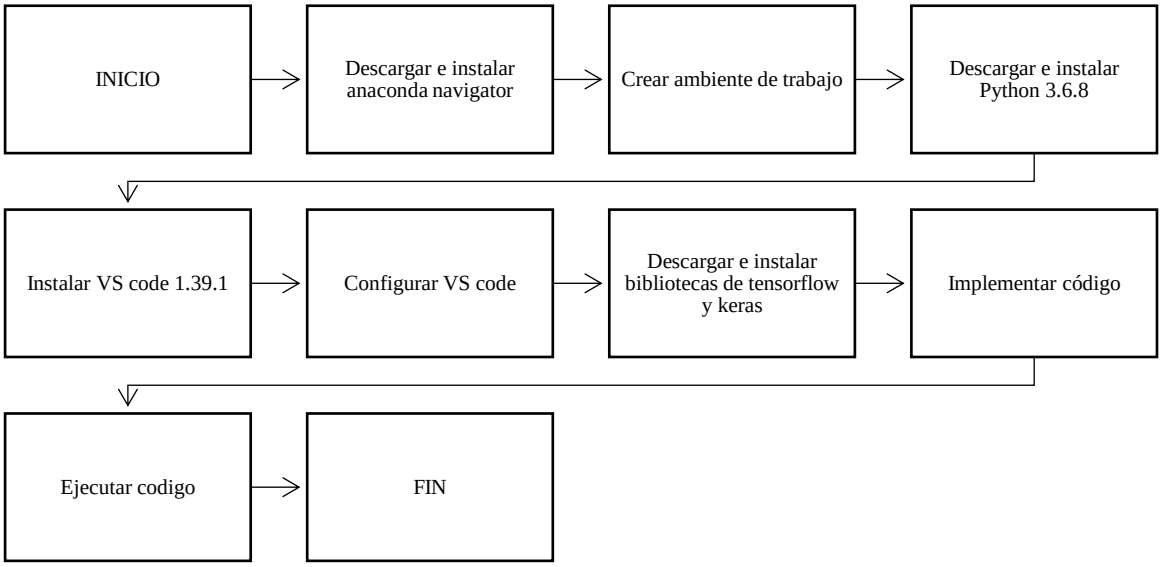

Figura 44. Pasos a seguir para implementar la red neuronal convolucional.

Luego que se instaló anaconda navigator, se abrió el terminal para poder realizar la creación y activación del ambiente de trabajo.

Comando para crear ambiente de trabajo:

conda create --name Nombre_del_ambiente

Comando para activar el ambiente de trabajo:

conda activate Nombre_del_ambiente

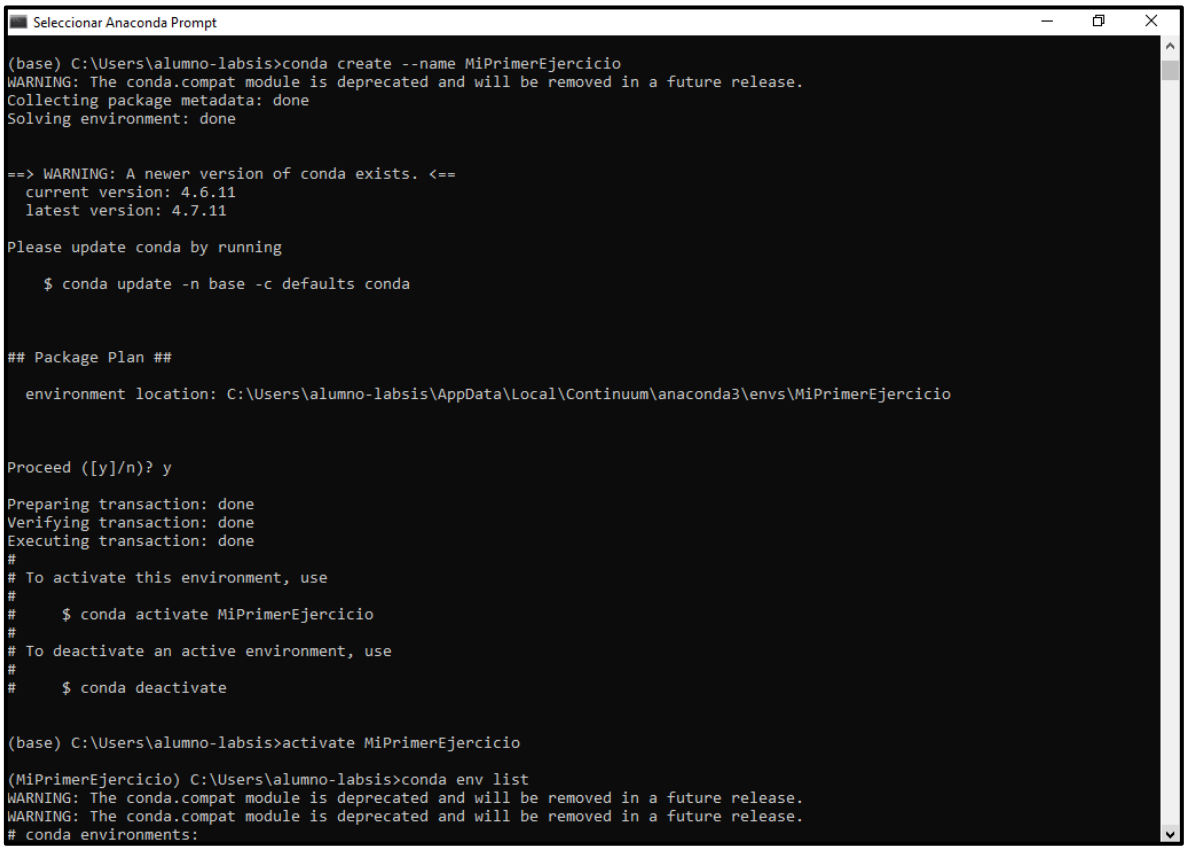

Figura 45. Creación y activación de ambiente de trabajo en anaconda prompt.

Para poder implementar la red neuronal convolucional se configuró el entorno de trabajo visual studio code, desactivando las opciones de python.linting.pylintEnable y python.linting.pep8Enabled ingresando a las configuraciones de visual studio (File > Preferences > Settings), está configuración permitió que al momento de implementar líneas de código en python no nos marcase como “error”.

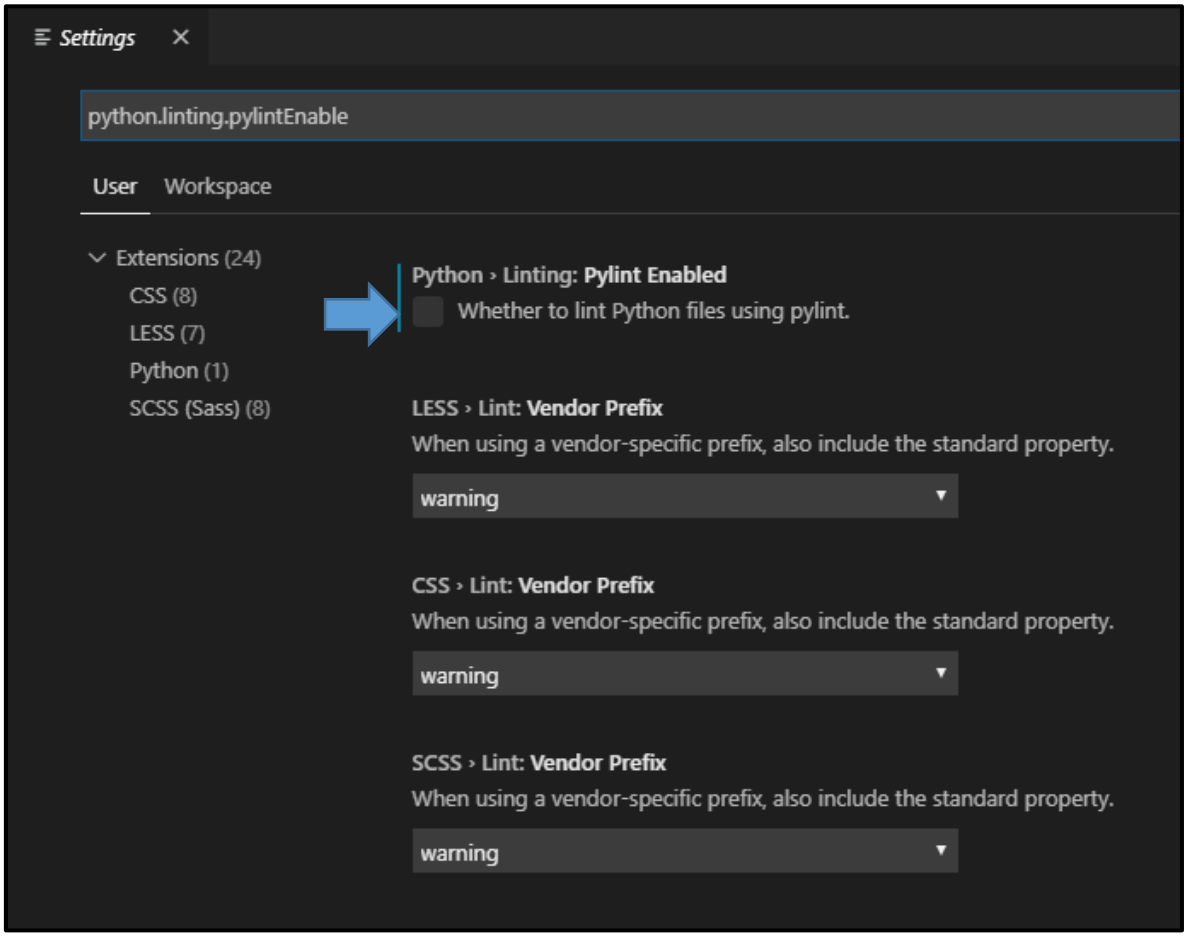

Figura 46. Configuración de visual studio code para poder implementar la red neuronal convolucional con python.

Luego que se activó el ambiente de trabajo y se configuró visual studio code, se realizó la instalación de la biblioteca tensorflow a través del comando "*pip install tensorflow*" porque está biblioteca es la que nos permitió y facilitó la implementación de la red neuronal convolucional.

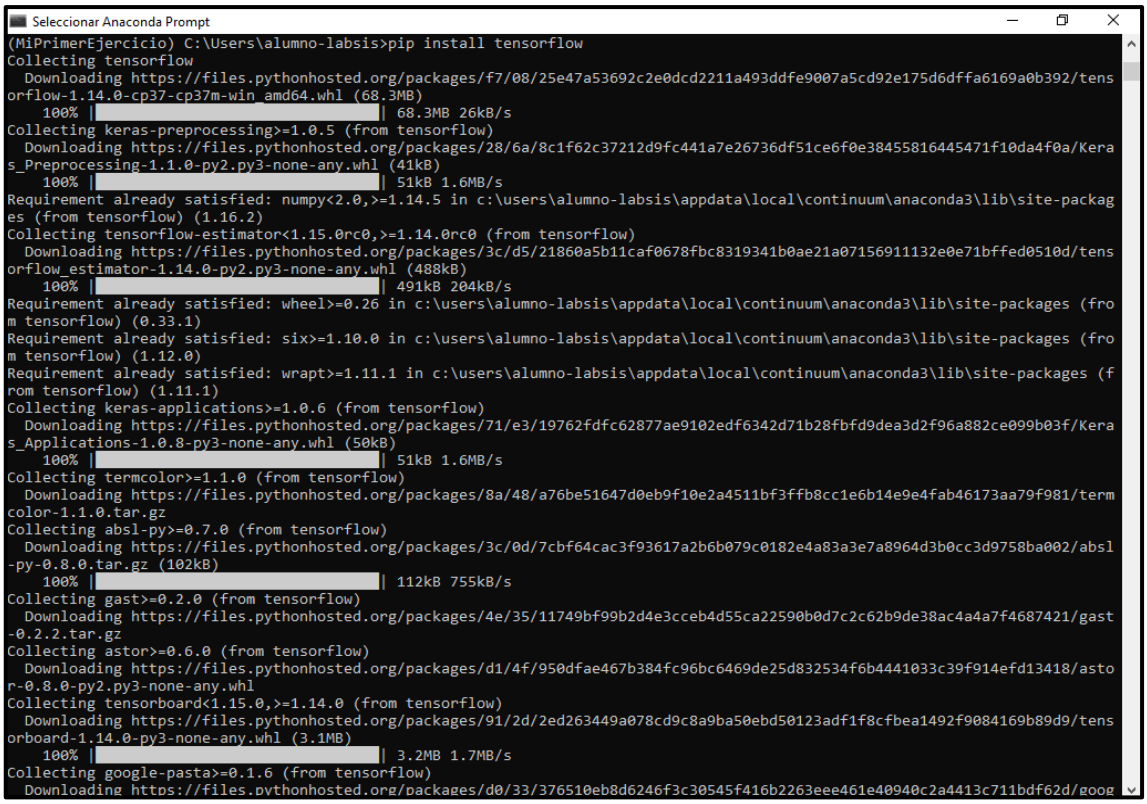

Figura 47. Instalación de biblioteca tensorflow.

Luego que ya se tiene el ambiente preparado se implementa el código de la red neuronal convolucional, primero importamos las librerías.

```
##Librerias para moverse en nuestro sistema operativos
import sys
import os
import numpy as np #Necesario para dibujar el historico de entrenamiento
import matplotlib.pyplot as plt
##Ayuda a preprocesar las imagenes para entrenar el algoritmo
from tensorflow.keras.preprocessing.image import ImageDataGenerator
##importamos optimizadores para entrenar el algoritmo
from tensorflow.keras import optimizers
##sequential es una librearia la cual nos permite hacer redes neuronales secuenciales es decir que cada una de las capas esta en orden
from tensorflow.keras.models import Sequential
##
from tensorflow.keras.layers import Dropout, Flatten, Dense, Activation
##capas para hacer una convolucion y max pooling
from tensorflow.keras.layers import  Convolution2D, MaxPooling2D
##nos ayuda a que si hay una sesion corriendo la cierra
from tensorflow.python.keras import backend as K

K.clear_session()
```

Figura 48. Importación de librerías.

Se crean dos variables que guarden la ruta de la capeta de entrenamiento y otra de validación.

```
##Parameters
epocas = 20
longitud, altura = 224, 224
batch_size = 32
pasos = 500 ##Pasos de entrenamiento
validation_steps = 100 ##pasos de validacion
filtrosConv1 = 32
filtrosConv2 = 64
filtrosConv3 = 128
tamano_filtro1 = (3, 3)
tamano_filtro2 = (2, 2)
tamano_filtro3 = (2, 2)
tamano_pool = (2, 2)
clases = len(os.listdir(data_entrenamiento))
lr = 0.0005
```

Figura 49. Asignación de parámetros que tuvo la red neuronal convolucional.

Antes de comenzar con la etapa de entrenamiento se realiza un pre procesamiento de las imágenes esto ayudará a que nuestra red neuronal convolucional no solo espere fotos en una sola posición.

```
#Preparamos (preprosesamos) nuestras imágenes
entrenamiento_datagen = ImageDataGenerator(
    rescale=1. / 255, #Le da a cada pixel el valor de 0 o 1 al momento de ser rescalado
    shear_range=0.2,#inclina nuestras imágenes
    zoom_range=0.2,#hace zoom a algunas imágenes
    horizontal_flip=True)##invierte las imágenes

test_datagen = ImageDataGenerator(rescale=1. / 255)
```

Figura 50. Código fuente para pre procesar imágenes.

Luego que se pre procesó las imágenes, estas imágenes se empiezan a utilizar.

```
entrenamiento_generador = entrenamiento_datagen.flow_from_directory(
    data_entrenamiento, #entra al directorio
    target_size=(altura, longitud),
    batch_size=batch_size,
    class_mode='categorical')

validacion_generador = test_datagen.flow_from_directory(
    data_validacion,
    target_size=(altura, longitud),
    batch_size=batch_size,
    class_mode='categorical')
```

Figura 51. Código fuente para utilizar imágenes pre procesadas.

Luego se procedió a implementar la red neuronal convolucional, con 3 capas de convolución seguida de una capa de maxpooling por cada capa de convolución, seguida de una capa aplanado, seguida de una capa totalmente conectada con 256 nodos, con una función de activación ReLu para evitar la no linealidad, seguido de una apagado aleatorio del 50%, y una capa de salida completamente conectada con 5 unidades que son las clases y una función de activación softmax.

```
#Crear la red neuronal convolucional
cnn = Sequential()
cnn.add(Convolution2D(filtrosConv1, tamano_filtro1, padding ="same", input_shape=(longitud, altura, 3), activation='relu'))
cnn.add(MaxPooling2D(pool_size=tamano_pool))

cnn.add(Convolution2D(filtrosConv2, tamano_filtro2, padding ="same", activation='relu'))
cnn.add(MaxPooling2D(pool_size=tamano_pool))

cnn.add(Convolution2D(filtrosConv3, tamano_filtro3, padding ="same", activation='relu'))
cnn.add(MaxPooling2D(pool_size=tamano_pool))

cnn.add(Flatten()) #aplanamos la informacion
cnn.add(Dense(256, activation='relu'))
cnn.add(Dropout(0.5)) #apagamos el 50% de las neuronas para no sobrecargar las neuronas, para que no aprenda un solo camino y cada vez sea aleatoria
cnn.add(Dense(clases, activation='softmax')) #esta capa indica el porcentaje de cada clase, sumando todos los valores da un total de 1
```

Figura 52. Código fuente para crear la estructura de la red neuronal convolucional.

Luego se procedió a compilar el modelo especificando la función de perdida de "*categorical crossentropy*" porque esta se utiliza para problemas de clasificación en los que solo un resultado puede ser correcto y como métrica se utilizó "*accuracy*" para medir la tasa de aciertos. También se procedió a realizar el entrenamiento del modelo indicando las épocas y pasos.

```
#compilamos el modelo
cnn.compile(loss='categorical_crossentropy',
            optimizer=optimizers.Adam(learning_rate=lr),
            metrics=['accuracy'])

#entrenamos la red neuronal
cnn.fit(
 entrenamiento_generador,
 steps_per_epoch=pasos,
 epochs=epocas,
 validation_data=validacion_generador,
 validation_steps=validation_steps)
```

Figura 53. Código fuente para compilar y entrenar la red neuronal convolucional.

Luego seleccionamos la carpeta donde se almacenará los 2 archivos que genera el entrenamiento (modelo.h5 y pesos.h5), en el caso que la carpeta no está creada se procede a crear.

```
carpeta = './modelo/'
if not os.path.exists(carpeta):
  os.mkdir(carpeta)
cnn.save('./modelo/modelo.h5')
cnn.save_weights('./modelo/pesos.h5')
```

Figura 54. Código fuente para almacenar archivos del entrenamiento y generar los gráficos exactitud y perdida de la etapa de entrenamiento.

Implementación de transferencia de aprendizaje con modelo VGG16

La estructura del código fuente es muy similar a la anterior, la diferencia es que, en lugar de nosotros crear la red neuronal convolucional y su estructura, utilizamos un modelo ya creado.

Se importan las librerías, y muy importante es importar la librería que contiene los modelos entrenados (“*applications*”).

```
##Librerias para moverse en nuestro sistema aoperativos
import sys
import os
import numpy as np #-------Necesario para dibujar el historico de entrenamiento
##Ayuda a preprosesar las imagenes para entrenar el algoritmo
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
##importamos optimizadores para entrenar el algoritmo
from tensorflow.python.keras import optimizers
##sequential es una librearia la cual nos permite hacer redes neuronales secuenciales es decir que cada una de las capas
#esta en orden
from tensorflow.python.keras.models import Sequential
##
from tensorflow.python.keras.layers import Dropout, Flatten, Dense, Activation
##capas para hacer una convolucion y max pooling
from tensorflow.python.keras.layers import  Convolution2D, MaxPooling2D
##nos ayuda a que si hay una sesion corriendo la cierra
from tensorflow.python.keras import backend as K
from tensorflow.python.keras import  applications #tiene modelos pre entrenados de keras
import matplotlib.pyplot as plt  #-------Necesario para dibujar el historico de entrenamiento

K.clear_session()
```

Figura 55. Importación de librerías.

Se crean dos variables que guarden la ruta de la capeta de entrenamiento y otra de validación.

```
data_entrenamiento = './gatos 224x224/entrenamiento'
data_validacion = './gatos 224x224/validacion'

epocas=20
#longitud, altura = 150, 150
longitud, altura = 224, 224 #dimension por defecto de VGG16
batch_size = 32
pasos = 500
##validation_steps = 300
validation_steps = 100
tamano_pool = (2, 2)
clases = 5
lr = 0.0004
```

Figura 56. Código fuente para asignar parámetros que tuvo la red neuronal convolucional.

Etapa de pre procesamiento de las imágenes, esto ayudará a que nuestra red neuronal convolucional no solo espere fotos en una sola posición.

```
##Preparamos nuestras imagenes

entrenamiento_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1. / 255)
```

Figura 57. Código fuente para pre procesar las imágenes.

Luego que se pre procesaron las imágenes, estas se proceden a utilizar.

```
entrenamiento_generador = entrenamiento_datagen.flow_from_directory(
    data_entrenamiento,
    target_size=(altura, longitud),
    batch_size=batch_size,
    class_mode='categorical')

validacion_generador = test_datagen.flow_from_directory(
    data_validacion,
    target_size=(altura, longitud),
    batch_size=batch_size,
    class_mode='categorical')
```

*Figura 58.*Código fuente para utilizar imágenes pre procesadas.

Se obtiene el modelo VGG16 y se elimina la última capa de salida porque este modelo tiene 1000 y solo necesitamos 5 para este trabajo que son las clases.

```
def modelo():
    vgg=applications.vgg16.VGG16()
    cnn=Sequential()
    for capa in vgg.layers:
        cnn.add(capa)
    cnn.layers.pop()
    for layer in cnn.layers:
        layer.trainable=False
    cnn.add(Dense(clases,activation='softmax'))
    return cnn

##CREAR LA RED VGG16
cnn=modelo()

cnn.compile(loss='categorical_crossentropy',
            optimizer=optimizers.Adam(lr=lr),
            metrics=['accuracy'])
```

Figura 59. Código fuente para utilizar el modelo VGG16.

Luego se procede a realizar el entrenamiento utilizando el modelo VGG16

```
#Creamos una variable h para dibujar el historial de entrenamiento
h = cnn.fit_generator(
    entrenamiento_generador,
    steps_per_epoch=pasos,
    epochs=epocas,
    validation_data=validacion_generador,
    validation_steps=validation_steps)
h.history
```

Figura 60. Código fuente para entrenar la red neuronal convolucional.

Luego seleccionamos la carpeta donde se almacenará los 2 archivos que genera el entrenamiento (modelo.h5 y pesos.h5), en el caso que la carpeta no está creada se procede a crear.

```
target_dir = './modelo/'
if not os.path.exists(target_dir):
  os.mkdir(target_dir)
cnn.save('./modelo/modeloVGG16.h5')
cnn.save_weights('./modelo/pesosVGG16.h5')

plt.figure(0)
plt.plot(h.history['acc'],'r')
plt.plot(h.history['val_acc'],'g')
plt.xticks(np.arange(0, epocas+1, 1))
plt.rcParams['figure.figsize'] = (10, 8)
plt.xlabel("Número de epocas")
plt.ylabel("Exactitud")
plt.title("Precisión de entrenamiento vs Precisión de validación")
plt.legend(['Entrenamiento','Validación'])

plt.figure(1)
plt.plot(h.history['loss'],'r')
plt.plot(h.history['val_loss'],'g')
plt.xticks(np.arange(0, epocas+1, 1))
plt.rcParams['figure.figsize'] = (10, 8)
plt.xlabel("Número de epocas")
plt.ylabel("Pérdida")
plt.title("Pérdida de entrenamiento vs Validación de Pérdida")
plt.legend(['Entrenamiento','Validación'])
plt.show()
```

Figura 61. Código fuente para almacenar archivos del entrenamiento y generar los gráficos exactitud y perdida de la etapa de entrenamiento.