Reconocimiento facial en perros, un modelo de vigilancia beneficioso

ELIZABETH AMADOR BASSAURE, VERÓNICA GONZÁLEZ CÁRDENAS, CARLOS ANTONIO MARTÍNEZ HERNÁNDEZ.

***Resumen*— Cuando se convive con un perro, las personas tienen que estar al pendiente de que salga al jardín a dar una vuelta, actuando de porteros abriendo y cerrando la puerta. El presente artículo describe un sistema de reconocimiento facial de mascotas que se puede implementar a puertas equipadas con una cámara que detecta a un perro y lo reconoce. Este sistema usa redes neuronales multicapa para reunir las características físicas más importantes de los perros que lo diferencian del resto, lo que permite que la puerta se abra solo para una mascota determinada.**

I. INTRODUCCIÓN

El uso del reconocimiento facial en animales ha demostrado tener mayor eficiencia que en los humanos.[1] En la mayoría de los casos, se utiliza para mejorar sus vidas con el propósito de preservar algunas especies y también por razones comerciales.

La aplicación ***GoGo Chicken,*** utiliza el reconocimiento facial para distinguir a los pollos, con el fin de que los consumidores puedan comprobar fácilmente el lugar de nacimiento, la alimentación y la información sanitaria del animal recién comprado, sin tener que recurrir a etiquetas o chips.[2]

En África, estos sistemas se utilizan para catalogar miles de ***leones*** y mantener su población bajo control. El monitoreo de ***elefantes***, por otro lado, facilita el descubrimiento de casos de caza furtiva. [3]

Investigadores de la Universidad George Washington ubicada en Washington D.C, Estados Unidos crearon ***LemurFaceID,*** un sistema de reconocimiento facial capaz de distinguir un lémur de otro con una precisión del 97%.[4]

PiP es una aplicación que explota el reconocimiento facial para encontrar mascotas perdidas. Se puede alertar de la desaparición en la aplicación, publicando una foto. La idea es que aquellos que lo encuentren puedan tomar una foto con PiP y así localizar al propietario. [5]

Entre las limitaciones que presentan estos proyectos se destacan por el uso que se le dé, ya que con la implementación del reconocimiento facial por parte del gobierno se está volviendo sistemático, hasta el punto de parecer un verdadero sistema de vigilancia masiva, en algunos casos violando la privacidad de las personas.

Elizabeth Amador Bassaure, Verónica González Cárdenas, Carlos Antonio Martínez Hernández pertenecen a la carrera de Ingeniería en Cibernética y Sistemas Computaciones de la Facultad de Ingeniería y realizaron el proyecto dentro del curso(s) Inteligencia Computacional (Email: carlos.martinezh96@hotmail.com, vero.6597@hotmail.com) El proyecto fue asesorado por: Roberto Antonio Vázquez de los Monteros

Los autores agradecen a: Roberto Antonio Vázquez de los Monteros, profesor de la materia de Inteligencia Computacional.

La ventaja que ofrece este tipo de tecnología puede ser realmente útil para hacer un seguimiento de los ecosistemas y gestionar las poblaciones. En este caso, el proyecto tiene la ventaja de mejorar las condiciones de vida, tanto para la mascota como para los dueños, al mismo tiempo ofrece una mayor seguridad en la casa ya que solo permitirá la entrada de una mascota determinada.

II. CONCEPTOS BÁSICOS

Algoritmo SIFT: Extrae los puntos clave y calcula sus descriptores. Para detectar esquinas más grandes se necesitan ventanas más grandes. Para esto, se utiliza el filtrado de espacio de escala. En él, se encuentra el laplaciano de Gauss para la imagen con varios σ valores. LoG actúa como un detector de manchas que detecta manchas en varios tamaños debido al cambio en σ. En resumen, σ actúa como un parámetro de escala. Por lo tanto, se pueden encontrar los máximos locales en la escala y el espacio que dé da una lista de (x, y, σ) valores, lo que significa que hay un posible punto clave en (x, y) en σ escala.

Una vez que se encuentra este DoG, se buscan imágenes de extremos locales sobre la escala y el espacio. Posteriormente, se deben encontrar las ubicaciones de puntos clave potenciales, se deben refinar para obtener resultados más precisos. Se utiliza la expansión de la escala de Taylor del espacio de escala para obtener una ubicación más precisa de los extremos, y si la intensidad en este extremo es menor que un valor de umbral (0,03 según el documento), se rechaza. Este umbral se llama contrastThreshold en OpenCV.

Los bordes deben eliminarse. Para ello, se utiliza un concepto similar al detector de esquinas Harris. Se usa una matriz hessiana de 2x2 (H) para calcular la curvatura principal. Sabemos por el detector de esquinas Harris que, para los bordes, un valor propio es más grande que el otro. Así que aquí se usa una función simple. Si esta relación es mayor que un umbral, llamado edgeThreshold en OpenCV, ese punto clave se descarta. Por lo tanto, elimina los puntos clave de bajo contraste y los puntos clave de borde y lo que queda son los fuertes puntos de interés.

Ahora se asigna una orientación a cada punto clave para lograr la invariancia a la rotación de la imagen. Se toma una vecindad alrededor de la ubicación del punto clave en función de la escala, y la magnitud y dirección del gradiente se calcula en esa región. Se crea un histograma de orientación con 36 compartimientos que cubren 360 grados (se evalúa por magnitud de gradiente y ventana circular con ponderación

gaussiana con σ igual a 1,5 veces la escala del punto clave). Se toma el pico más alto en el histograma y cualquier pico por encima del 80% también se considera para calcular la orientación. Crea puntos clave con la misma ubicación y escala, pero con diferentes direcciones. Contribuye a la estabilidad de emparejamiento.

Ahora se crea el descriptor de punto clave. Se toma una vecindad 16x16 alrededor del punto clave. Se divide en 16 sub-bloques de tamaño 4x4. Para cada sub-bloque, se crea un histograma de orientación de 8 casillas. Así que un total de 128 valores de bin están disponibles. Se representa como un vector para formar un descriptor de punto clave. Además de esto, se toman varias medidas para lograr robustez contra los cambios de iluminación, rotación, etc.

Los puntos clave entre dos imágenes se combinan mediante la identificación de sus vecinos más cercanos. Pero en algunos casos, la segunda coincidencia más cercana puede estar muy cerca de la primera. Puede suceder debido al ruido u otras razones. En ese caso, se toma la relación de la distancia más cercana a la segunda distancia más cercana. Si es mayor que 0,8, se rechazan. Elimina alrededor del 90% de las coincidencias falsas, mientras que descarta solo el 5% de las coincidencias correctas, según el documento.

[6] Imagen 1: Ejemplo resultado de usar algoritmo SIFT.

Algoritmo SURF: SIFT, se aproximó al laplaciano de gaussiano con la diferencia de gaussiano para encontrar escala-espacio. SURF va un poco más lejos y se aproxima a LoG con filtro de caja. Una gran ventaja de esta aproximación es que la convolución con filtro de caja se puede calcular fácilmente con la ayuda de imágenes integrales. Y se puede hacer en paralelo para diferentes escalas. También el SURF depende del determinante de la matriz de Hess para la escala y la ubicación.

Para la asignación de orientación, SURF usa respuestas wavelet en dirección horizontal y vertical para un vecindario de tamaño 6s. También se le aplican pesos guassianos adecuados. Luego se trazan en un espacio, la orientación dominante se calcula con la suma de todas las respuestas dentro de una ventana de orientación deslizante de ángulo de 60 grados. La respuesta de wavelet se puede encontrar utilizando imágenes integrales muy fácilmente en cualquier escala. Para muchas aplicaciones, la invarianza de rotación no es necesaria, por lo que no es necesario encontrar esta orientación, lo que acelera el proceso.

SURF proporciona una funcionalidad llamada Upright-SURF o U-SURF. Mejora la velocidad y es robusto. OpenCV soporta ambos, dependiendo de la bandera, en posición vertical. Si es 0, se calcula la orientación. Si es 1, la orientación no se calcula y es más rápida.

Para la descripción de la función, SURF usa las respuestas de Wavelet en dirección horizontal y vertical (nuevamente, el uso de imágenes integrales facilita las cosas). Se toma una vecindad de tamaño 20sX20s alrededor del punto clave donde s es el tamaño. Se divide en 4x4 subregiones. Para cada subregión, se toman respuestas de wavelets horizontales y verticales y se forma un vector como este v = (\ sum {d_x}, \ sum {d_y}, \ sum {| d_x |}, \ sum {| d_y |}).

Para mayor distinción, el descriptor de características SURF tiene una versión ampliada de 128 dimensiones. Las sumas de d_xy | d_x |se calculan por separado para d_y <0y d_y \ geq 0. De manera similar, las sumas de d_yy | d_y |se dividen de acuerdo con el signo de d_x, duplicando así el número de entidades. No agrega complejidad de computación. OpenCV admite ambos estableciendo el valor del indicador extendido con 0 y 1 para 64-dim y 128-dim respectivamente (el valor predeterminado es 128-dim)

El signo del laplaciano distingue las manchas brillantes en fondos oscuros de la situación inversa. En la etapa de emparejamiento, solo comparamos las funciones si tienen el mismo tipo de contraste (como se muestra en la imagen a continuación). Esta información mínima permite una comparación eficiente sin reducir el rendimiento del descriptor.

En resumen, SURF agrega muchas características para mejorar la velocidad en cada paso. El análisis muestra que es 3 veces más rápido que SIFT, mientras que el rendimiento es comparable a SIFT. SURF es bueno para manejar imágenes con desenfoque y rotación, pero no es bueno para manejar el cambio de punto de vista y el cambio de iluminación.

[7] Imagen 2: Ejemplo resultado de usar algoritmo SURF.

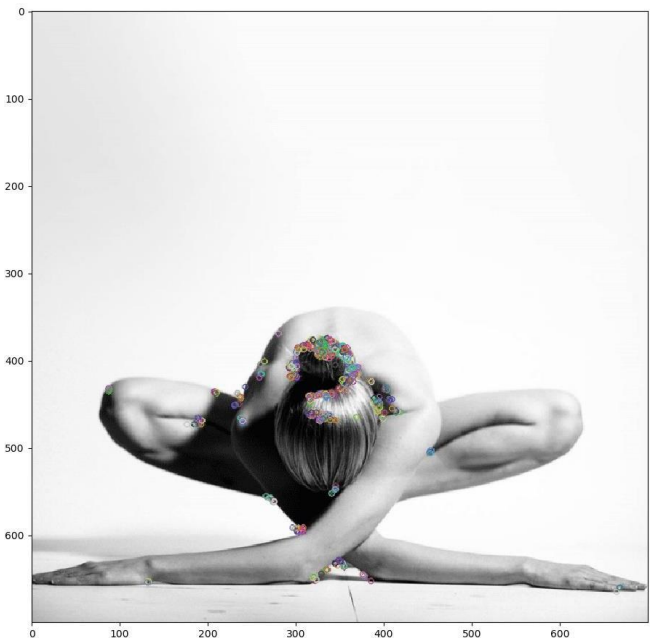
Algoritmo ORB: Alternativa a SIFT y SURF en computación Coste, igualar el rendimiento y sobre todo las

patentes. ORB es básicamente una fusión del detector de puntos clave FAST y el descriptor BREVE con muchas modificaciones para mejorar el rendimiento.

Primero se usa FAST para encontrar puntos clave, luego sea aplica la medida de la esquina de Harris para encontrar los mejores N puntos entre ellos. También se usó pirámide para producir características multiescala. Pero un problema es que FAST no calcula la orientación. A los autores se les ocurrió la siguiente modificación: Calcula la intensidad ponderada del centroide del parche con la esquina ubicada en el centro. La dirección del vector desde este punto de esquina hasta el centroide da la orientación. Para mejorar la invariancia de rotación, los momentos se calculan con x e y que deben estar en una región circular de radio, donde es el tamaño del parche.

ORB usa descriptores BRIEF. Lo que ORB hace es dirigir BRIEF según la orientación de los puntos clave. Para cualquier conjunto de características de pruebas binarias en la ubicación (x_i, y_i), defina una matriz que contenga las coordenadas de estos píxeles. Luego, utilizando la orientación del parche, se encuentra su matriz de rotación y gira para obtener la versión dirigida (girada).

ORB discretiza el ángulo en incrementos de (12 grados), y construye una tabla de búsqueda de patrones BRIEF precomputados. Siempre que la orientación de punto clave coherente en todas las vistas, se utilizará el conjunto correcto de puntos para calcular su descriptor. ORB realiza una búsqueda entre todas las pruebas binarias posibles para encontrar las que tienen una varianza alta y un valor cercano a 0,5, además de no estar correlacionados. El resultado se llama rBRIEF. ORB es más eficiente que SURF y el descriptor SIFT y ORB funciona mejor que SURF. ORB es una buena opción en dispositivos de baja potencia para costura panorámica, etc.



[8] Imagen 3: Ejemplo resultado de usar algoritmo ORB.

Red neuronal multicapa: La red multicapa es una red de alimentación hacia adelante (feedforward) compuesta por una capa de unidades de entrada (sensores), otra capa de unidades de salida y un número determinado de capas intermedias de unidades de proceso, también llamadas capas ocultas porque no se ven las salidas de dichas neuronas y no tienen conexiones con el exterior.

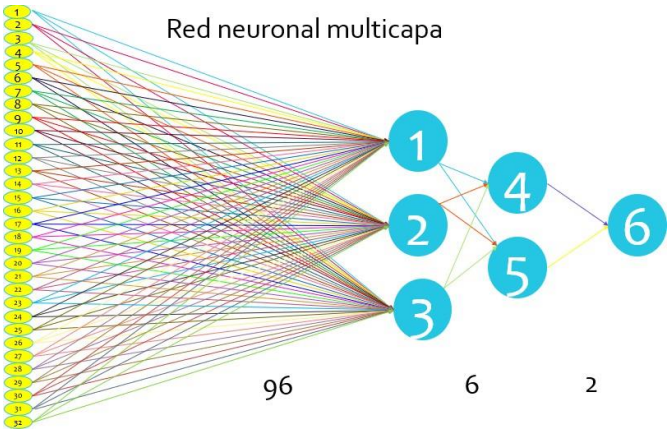
Cada sensor de entrada está conectado con las unidades de la segunda capa, y cada unidad de proceso de la segunda capa está conectada con las unidades de la primera capa y con las unidades de la tercera capa, así sucesivamente. Las unidades de salida están conectadas solamente con las unidades de la última capa oculta.

Con esta red se pretende establecer una correspondencia entre un conjunto de entrada y un conjunto de salidas. Para ello se dispone de un conjunto de patrones p de entrenamiento, de manera tal que se sabe que al patrón de entrada le corresponde la salida. Es decir, se conoce dicha correspondencia para p patrones.

$$y_{ji} = f\left(\sum_{k=1}^{N_{j-1}} y_{(j-1)k} w_{jki}\right)$$

[9] **Ecuación para calcular la salida de una neurona.**

[10] Imagen de la Red Neuronal utilizada en el proyecto.



Algoritmo de retro propagación: Se trata de determinar los pesos de las conexiones sinápticas entre las unidades de proceso de manera que las salidas de la red coincidan con las salidas deseadas, o por lo menos, sean próximas. Es decir, se trata de determinar los pesos de manera que el error total sea mínimo.

El algoritmo de retropropagación utiliza el método del descenso por el gradiente y realiza un ajuste de los pesos comenzando por la capa de salida, según el error cometido, y se procede propagando el error a las capas anteriores, de atrás hacia delante, hasta llegar a la capa de las unidades de entrada. Una característica importante de este algoritmo es su capacidad para organizar el conocimiento de la capa intermedia de manera que se pueda conseguir cualquier correspondencia entre la capa de entrada y la de salida obtenida con la deseada.

A continuación, se calcula el error para cada neurona de salida, y este valor de error es transmitido hacia atrás, por

todas las capas intermedias, y se van modificando sus pesos sinápticos según dicho error y los valores de las salidas de las unidades de proceso precedentes ponderados por sus pesos sinápticos.

$$w_{jki}(t+1) = w_{jki} + \gamma\delta_{ji}\frac{df(x)}{dx}y_{(j-1)k}$$

11 Ecuación para calcular los pesos sinápticos.

III. METODOLOGÍA PROPUESTA

Resumen etapas consideradas en la metodología: El proyecto utiliza las librerías de OpenCV para la interpretación de imágenes. Funciona para C/C++, Java, Phyton, entre otras por ser una librería completa en funciones para realizar operaciones sobre imágenes. Posteriormente, se crea la base de datos con las imágenes seleccionadas y las características únicas de cada una. Después se inicia el entrenamiento del algoritmo al obtener los pesos sinápticos y guárdalos para identificar al perro. Finalmente, se evalúan las imágenes y se realizan varias pruebas con diferentes imágenes de los perros seleccionados para garantizar el aprendizaje del algoritmo.

12 Diagrama - Etapas de la metodología.

Descripción detallada de las etapas consideradas en la metodología:

ETAPA 1 – MANEJO DE IMÁGENES:

En primer lugar, se debe usar el ***algoritmo SIFT,*** se inicia con la detección de puntos clave de una imagen.

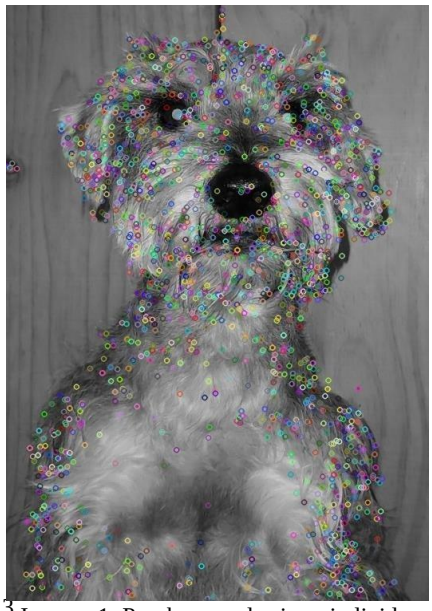
Primero se construye un objeto SIFT. Se introducen diferentes parámetros que son opcionales.

```
importar numpy como np
importar cv2 como cv
img = cv.imread ( 'home.jpg' )
gris = cv.cvtColor (img, cv.COLOR_BGR2GRAY)
sift = cv.xfeatures2d.SIFT_create ()
kp = sift.detect (gris, Ninguno )
img = cv.drawKeypoints (gris, kp, img)
cv.imwrite ( 'sift_keypoints.jpg' , img)
```

La función **sift.detect()** encuentra el punto clave en las imágenes. Puede pasar una máscara si se desea buscar solo una parte de la imagen. Cada punto clave es una estructura especial que tiene atributos como sus coordenadas (x, y), el tamaño del vecindario significativo, el ángulo que especifica su orientación, la respuesta que especifica la fuerza de los puntos clave, etc.

OpenCV también proporciona la función **cv.drawKeyPoints** que dibuja los pequeños círculos en las ubicaciones de los puntos clave. Si pasa una bandera, dibujará un círculo con el tamaño del punto clave e incluso mostrará su orientación.

```
img = cv.drawKeypoints (gris, kp, img, flags
= cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv.imwrite ( 'sift_keypoints.jpg' , img)
```



13 Imagen 1: Prueba con el primer individuo.

Ahora para calcular el descriptor, OpenCV proporciona dos métodos.

1. Como ya encontró puntos clave, puede llamar a **sift.compute** que calcula los descriptores de los puntos clave que se han encontrado.
2. Si no encontró puntos clave, busque directamente puntos clave y descriptores en un solo paso con la función, **sift.detectAndCompute.**

```
sift = cv.xfeatures2d.SIFT_create ()
kp,  des  =  sift.detectAndCompute
     (gris, Ninguno)
```

En segundo lugar, se debe usar el ***algoritmo SURF***, se inicia un objeto SURF con algunas condiciones como descriptores de 64/128-dim, SURF vertical / vertical, etc.

Luego, se usa **SURF.detect**, **SURF.compute**, etc. para encontrar puntos clave y descriptores. Demostración simple de cómo encontrar los puntos clave y los descriptores de SURF y cómo dibujarlos.

```
>>> img = cv.imread ( 'fly.png' , 0)
# Crear objeto SURF. Puede especificar
params aquí o más tarde.
# Aquí pongo el umbral de Hessian a 400
```

```
>>> surf = cv.xfeatures2d.SURF_create
(400) # Buscar puntos clave y descriptores
directamente
>>> kp, des = surf.detectAndCompute
(img, Ninguno )
>>> len
(kp) 699
```

Finalmente, se dibuja en la imagen y verificamos el tamaño del descriptor y se cambia a 128 si es solo 64-dim.

```
>>> img2 = cv.drawKeypoints (img, kp, Ninguno
, (255,0,0), 4)
>>> plt.imshow (img2), plt.show ()
# Encuentra el tamaño del descriptor
>>> imprimir (surf.descriptorSize ())64
# Eso significa bandera, "extendido" es Falso.
>>> surf.getExtended () Falso
# Así que hacemos True para
obtener descriptores de 128 dim.
>>> surf.setExtended ( True )
>>> kp, des = surf.detectAndCompute
(img, Ninguno )
>>> imprimir (surf.descriptorSize ()) 128
>>> imprimir (des.shape) (47, 128)
```

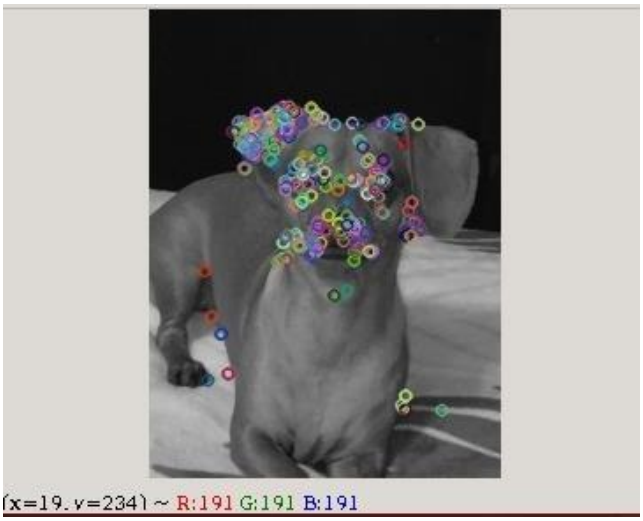
En tercer lugar, se usa el ***algoritmo ORB***, se tiene que crear un objeto ORB con la función, cv2.ORB () o utilizando la interfaz común feature2d. Los parámetros optimizados son nFeatures que indican el número máximo de características a retener (por defecto 500), scoreType que indica si la puntuación de Harris o FAST para clasificar las características (por defecto, la puntuación de Harris) etc. Otro parámetro, WTA_K decide el número de puntos que producen cada elemento del descriptor BRIEF orientado. Por defecto son dos, es decir, selecciona dos puntos a la vez. En ese caso, para la comparación se utiliza la distancia NORM_HAMMING. Si WTA_K es 3 o 4, lo que toma 3 o 4 puntos para producir el descriptor BRIEF, entonces la distancia de coincidencia es definida por NORM_HAMMING2.

```
import numpy as
np import cv2
from matplotlib import pyplot as
plt img = cv2.imread('ojo.jpg',0)
# Initiate STAR
detector orb =
cv2.ORB_create()
# find the keypoints with
ORB kp =
orb.detect(img,None)
# compute the descriptors with
ORB kp, des = orb.compute(img,
kp)
# draw only keypoints location,not size
and orientation
img2 = cv2.drawKeypoints(img,kp,img,
flags=0) plt.imshow(img2),plt.show()
```

Para este proyecto, se decidió usar ORB ya que tiene menos puntos. Con SIFT y SURF se obtienen cerca de 2000 puntos de 140 características cada uno, con ORB se obtiene 345 puntos de 32 características cada uno.

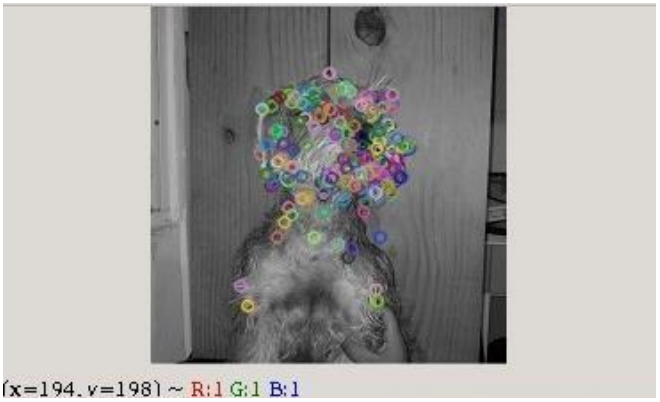
ETAPA 2 – BASE DE DATOS:

Se toma las 20 imágenes y se insertan en la base de datos (en este caso, un documento .txt) donde en promedio se obtiene de 300 a 400 puntos por imagen.



[14]Imagen 2: Sujeto de prueba “Camila”. Valores de los colores.

En un documento de extensión txt se guardan los vectores obtenidos en el análisis de la imagen. De los 345 vectores, se realiza un promedio para obtener un vector de 32 por imagen. Se extrae toda la información del documento y se coloca en un vector de 640 lugares (20 imágenes, 10 imágenes por perro * 32 características cada uno). Con esto sabemos que cada 32 datos es una imagen, por lo tanto, se coloca en un matriz por renglón esos 32 datos y el 33 es el deseado. (Para el perro de muestra “Rocky” junto con sus 10 imágenes es 0, para el perro de muestra “Camila” junto con sus 10 imágenes es 1.)



[15] Imagen 3: Sujeto de prueba “Rocky”. Valores de los colores.

ETAPA 3 – ENTRENAMIENTO DEL ALGORITMO:

Se decidió realizar una red multicapa de 3 capas para una mejor precisión y repartir la carga de trabajo en 6 neuronas. Con ayuda del algoritmo de retropropagación se adquirieren los valores de las 6 neuronas.

Se obtuvieron los errores, y se evaluó el criterio de paro que fue 0.01, ya que se necesita un mayor grado de precisión. Si se obtiene que el error es menor al criterio de paro, los últimos pesos sinápticos (w) se guardan en un documento

(.txt) para posteriormente evaluar utilizando esos pesos sinápticos.

ETAPA 4 - EVALUACIÓN DE LA IMÁGEN:

Se extrae la información del documento que contiene a los pesos sinápticos para realizar el procedimiento de entrenar al algoritmo de retro propagación, pero sin realizar ninguna modificación a los pesos sinápticos. Posteriormente, se obtiene el error y se compara con los obtenidos en el paso dos para identificar a que perro corresponde la imagen, o en su defecto mandar un mensaje de que el perro no fue detectado.

IV. Resultados.

Se contó con 10 fotos del perro Rocky y 10 fotos del perro Camila, en las pruebas de código, se tuvo que dividir el promedio de los puntos obtenidos de la foto entre 1000 ya que los números contaban con más de 5 cifras, se eliminó de la función 1/1+e^-x, ya que la x tiene un valor muy grande lo que ocasiona desbordamientos y el de error siempre se obtenía 1 e impedía cambiarlo ya que el promedio siempre daba de resultado 0.5. El programa variará en el número de épocas, a veces serán 100 épocas, otras 500 épocas. En la fase de prueba con cada foto, el algoritmo fue acertado en un 80% de las veces, ya que tuvo un porcentaje de error bajo al probar fotos de cuerpo completo o con diferente ángulo. Cuando las fotos coinciden en espacio y forma, si se trata de otra raza diferente o incluso de la misma raza, pero diferente perro, el algoritmo detecta que no es el mismo que el aprendido y no libera el acceso.

V. Conclusiones.

Dependiendo de las necesidades del programa que se quiera realizar, se debe adaptar diferentes funciones de aprendizaje para la red neuronal. Aunque el algoritmo se tarda un tiempo y se debe de realizar un gran número de pruebas para verificar su funcionamiento, se garantiza el aprendizaje de la red neuronal en un 80%, es decir, “aprendió” a identificar las características que hacen único a cada perro en condiciones ideales.

En un futuro, se seguirá mejorando el algoritmo para reducir el porcentaje de error al mínimo y tener un algoritmo confiable para que se pueda implementar este sistema a un dispositivo con una puerta electrónica que permita salir al perro cuando se acerque a esta, mejorando la calidad de vida tanto de la mascota, como del dueño, ya que permite tener una mejor seguridad en la casa e impide la entrada a visitas no deseadas y la mascota puede entrar y salir del hogar cuando lo necesite y quiera.

Sujeto	Pruebas	# de error	% de error
Rocky	10	3	30%
Camila	10	2	20%
Otro perro	10	0	0%
Rocky peludo	10	0	0%

Sujeto Identificado	Error Camila	Error Rocky
Otro perro	-0.1261	-0.2261
Rocky	0.1130	0.0130
Camila	0.0389	-0.0610

```
Python 3.5.2 (default, Nov 12 2018, 13:43:14)
[GCC 5.4.0 20160609] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: /home/carlos/Documentos/prueba4.py ================
introduce 1 si deseas calcular la base o 2 si deseas calcular pesos 0 3 si desea
s analizar foto

cont 330
error camila -0.12610803803725368
error rocky -0.22610803803725377
OJO es otro perro
>>>
================ RESTART: /home/carlos/Documentos/prueba4.py ================
introduce 1 si deseas calcular la base o 2 si deseas calcular pesos 0 3 si desea
s analizar foto

cont 419
error camila 0.11302200503428828
error rocky 0.013022005034288187
es rocky
>>>
================ RESTART: /home/carlos/Documentos/prueba4.py ================
introduce 1 si deseas calcular la base o 2 si deseas calcular pesos 0 3 si desea
s analizar foto

cont 334
error camila 0.03892617479274918
error rocky -0.06107382520725091
es camila
>>>
```

1/ Imagen Resultados.

VI. Referencias

[1] Alberto Barbieri, A. (2018, 21 noviembre). El reconocimiento facial en animales, una tecnología beneficiosa. Recuperado 25 junio, 2019. Consultado, 02 de junio de 2019. Disponible en: https://www.nobbot.com/futuro/reconocimiento-facial-animales/

[2] Por Equipo de redacción, E. R. (2018, 5 octubre). Una compañía china podría desarrollar un sistema de reconocimiento facial para pollos ecológicos - Avicultura. Consultado, 02 de junio de 2019. Disponible en: https://avicultura.com/una-compania- china-desarrolla-un-sistema-de-reconocimiento-facial-para-pollos-ecologicos/

[3] Alberto Barbieri, A. (2018, 21 noviembre). El reconocimiento facial en animales, una tecnología beneficiosa. Recuperado 25 junio, 2019. Consultado, 02 de junio de 2019. Disponible en: https://www.nobbot.com/futuro/reconocimiento-facial-animales/

[4] David Crouse, D. C. (s.f.). LemurFaceID: a face recognition system to facilitate individual identification of lemurs. Consultado, 02 de junio de 2019. Disponible en: https://bmczool.biomedcentral.com/articles/10.1186/s40850-016-0011-9

[5] Pip. (2015, 13 febrero). PiP | The Pet Recognition App Consultado, 02 de junio de 2019. Disponible en: http://www.petrecognition.com/

[6] Gonzalo Pajares Martinsanz, Jesús M, de la Cruz García. Visión por computador: imágenes digitales y aplicaciones. J. 2010. (En Línea). Consultado, 02 de junio de 2019. Disponible en: http://es.scribd.com/doc/52002261/Imageem-1

[7] Galván, I y Valls, J. 2010. Redes de Neuronas Artificiales. (En Línea). Consultado, 27 de Mayo 2019. Formato (PDF). Disponible en: http://ocw.uc3m.es/ingenieria-informatica/redes-de-neuronas-artificiales/transparencias/material-de-clase.-tema-2

[8] Torres, Luis .2008. Redes Neuronales Artificiales. (En Línea).Consultado 30 de Mayo 2019Formato (PDF). Disponible en http://disi.unal.edu.co/~lctorress/RedNeu/RNA006c.pdf

[9] Valls, J. 2007. Redes de Neuronas Perceptrón y Adaline. (En Línea). Consultado, 30 de Mayo 2019. Formato (PDF).

Disponible en: http://eva.evannai.inf.uc3m.es/et/docencia/rn-inf/documentacion/Tema2-PerceptronAdaline.pdf

[10] Roncagllolo, P. s.f. Procesamiento Digital de Imágenes. (En Línea). Consultado, 01 de Junio 2019. Formato (PDF). Disponible en: http://www2.elo.utfsm.cl/~elo328/PDI21_RedesNeuronales.pdf