REVISTA INCAING
ISSN 2448 9131

(Marzo-Abril 2021), pp. 22-28

Algoritmo de reconocimiento facial para la detección de personas, basado en redes neuronales convolucionales y aprendizaje profundo.

Armando Sánchez Cuevas.
Anahí Rosas Pérez.
TecNM Campus Tepeaca
Departamento de Ciencias Básicas
Puebla, México.

armando.sc@tepeaca.tecnm.mx
eng.armando.cuevas@ieee.org

Resumen.

El uso y aplicación de inteligencia artificial, redes neuronales y software especializado como MATLAB ha permitido la codificación de un algoritmo que permite la identificación de personas utilizando diferentes tipos de cámaras. En este caso, se han utilizado tres tipos diferentes de cámaras para realizar la tarea específica de identificar personas.
También se utilizaron una cámara web, una cámara de tablero y una cámara de marcha atrás para uso automotriz. El resultado ha sido satisfactorio; Fue posible identificar a una persona específica utilizando el algoritmo basado en MATLAB mencionado en esta investigación.

Palabras clave: *IA, MATLAB, redes neuronales, cámaras.*

Abstract.

The use and application of artificial intelligence, neural networks and specialized software such as MATLAB has allowed the coding of an algorithm that allows the identification of people using different types of cameras. In this case, three different types of cameras have been used to perform the specific task of identifying people.
A webcam, dash cam, and reversing camera were also used for automotive use. The result has been satisfactory; It was possible to identify a specific person using the MATLAB-based algorithm mentioned in this research.

Keywords: *AI, MATLAB, neural networks, cameras.*

I. INTRODUCCIÓN

Las redes neuronales existen desde 1940 y, como resultado, tienen bastante historia. Un buen ejemplo de este avance histórico es la función de activación, que escala los valores que pasan por las neuronas de la red neuronal.

Una red neuronal del tipo: *"feedforward"* con muchas capas se convierte en una red neuronal profunda. La mayoría de los investigadores que hacen uso de esta tecnología consideran que las redes neuronales son un tipo de cerebro artificial. Según este punto de vista, las redes neuronales podrían impulsar robots, drones o sistemas expertos o mantener conversaciones inteligentes con seres humanos.

II. CONCEPTOS BÁSICOS SOBRE EL DESARROLLO DE LA PROPUESTA.

Redes neuronales.

Las redes neuronales son una pequeña parte de la IA. Tal como existen actualmente, las redes neuronales llevan a cabo tareas minúsculas y muy específicas. A diferencia del cerebro humano, las redes neuronales basadas en computadora no son dispositivos computacionales de propósito general. Además, el término red neuronal puede crear confusión porque el cerebro es una red de neuronas al igual que la IA utiliza redes neuronales [1].

Entrenamiento de redes neuronales: supervisado y no supervisado.

Cuando se especifica la salida ideal de una red neuronal, se usa un entrenamiento supervisado. Si no se proporcionaron de entrada los resultados ideales, se estaría entonces utilizando un entrenamiento neuronal sin supervisión. El entrenamiento supervisado enseña a la red neuronal a producir la salida ideal o la que mejor se adecua a nuestras necesidades. En este contexto entonces el entrenamiento no supervisado generalmente se encarga de enseñar o entrenar a la red neuronal y también a colocar los datos de entrada en varios grupos definidos por el recuento o retorno de las neuronas de salida. [2].

Métodos de aprendizaje profundo.

El uso y desarrollo de métodos exitosos de aprendizaje profundo en problemas de visión por computadora (es decir, clasificación y segmentación) [3]; ha motivado a la comunidad científica a trabajar ampliamente en técnicas de análisis de imágenes y también a investigar la aplicabilidad de tales métodos en problemas basados en robótica, segmentación de imágenes y problemas de clasificación de objetos, por ejemplo. Comparados con los métodos tradicionales de entrenamiento de redes neuronales ya aplicados anteriormente; en este caso el aprendizaje profundo ofrece las siguientes ventajas: aprendizaje automático de características estimadas en cuestión de funciones específicas de detección (segmentación y/o clasificación objetivo, por ejemplo); oportunidad de construir sistemas completos "de extremo a extremo" que toman una imagen, detectan, segmentan y clasifican objetos visuales (por ejemplo, personas, objetos, animales, edificios, etc.) utilizando modelos únicos y un proceso de capacitación unificado. [4].

Aspectos importantes del uso de redes neuronales convolucionales.

Los conjuntos de datos de imágenes definidas en conjuntos a gran escala (es decir, ImageNet [5] junto con las redes neuronales convolucionales profundas (CNN) [6]; han conducido a un rápido y exponencial progreso en el reconocimiento de imágenes naturales. Desde la perspectiva del aprendizaje basado en datos, los conjuntos de datos etiquetados a gran escala con características de distribución de datos representativas son cruciales para aprender modelos precisos o generalizables [6]. El paquete computacional llamado *ImageNet* por ejemplo, ofrece una base de datos muy completa de más de 1,2 millones de imágenes naturales categorizadas en más de 1000 clases.
Los modelos de CNN previamente entrenados en *ImageNet* sirven como la columna vertebral para muchos métodos de detección de objetos y segmentación de imágenes que están ajustados para otros conjuntos de datos. [1].

Dentro de una red neuronal profunda con Matlab.

Una red neuronal profunda definida en Matlab, combina múltiples capas de procesamiento no lineales, utilizando elementos simples que operan en paralelo e inspirados por sistemas nerviosos biológicos. Esta red neuronal consta de una capa de entrada, varias capas ocultas y una capa de salida. Estas capas están interconectadas a través de nodos o neuronas, y cada capa oculta utiliza la salida de la capa anterior como entrada. [7].

La figura 1 muestra el concepto anterior como ejemplo.

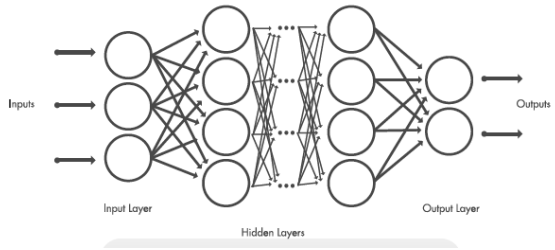

Fig. 1. Un ejemplo de una red neuronal profunda. [7].

Los dos tipos más comunes de aplicación del aprendizaje supervisado son la clasificación y la regresión. Estas palabras pueden parecer desconocidas, pero en realidad no son tan desafiantes.

En el primer caso se habla de la clasificación. Esta puede ser la aplicación más predominante de la técnica de aprendizaje basada en *Machine Learning*. El problema de clasificación se centra en encontrar literalmente las clases a las que pertenecen los datos.

Algunos ejemplos son los siguientes:

Servicio de filtrado de correo spam ➔ Clasifica los correos en regulares o spam.

Servicio de reconocimiento de dígitos ➔ Clasifica la imagen de dígitos de uno en uno por ejemplo del dígito 0 al dígito 9.

Servicio de reconocimiento facial ➔ Clasifica las imágenes provenientes de rostros de personas y en el caso de la definición o reconocimiento de usuarios registrados y no registrados.

Es preciso mencionar que el aprendizaje supervisado requiere pares de entrada y salida correctos para los datos de entrenamiento. De manera similar, los datos de entrenamiento del problema de clasificación usando MATLAB se codificaría así [7]:

{input, class}

En el problema de clasificación, queremos saber a qué clase pertenece la entrada. Entonces, el par de datos tiene la clase en lugar de la salida correcta correspondiente a la entrada [8].

Cómo aprende una red neuronal profunda.

Cuando se tiene un conjunto de imágenes donde cada imagen contiene una de cuatro categorías diferentes de objeto, y queremos que la red de aprendizaje profundo reconozca automáticamente qué objeto está en cada imagen. Es necesario realizar un etiquetado de las imágenes para tener datos de entrenamiento para la red. Con estos datos de entrenamiento, la red puede comenzar a aprender o comprender las características específicas del objeto previamente definido y así poder asociarlas con la categoría correspondiente.

En este caso cada capa de la red toma datos de la capa anterior, los transforma y los transmite. La red aumenta la complejidad y el detalle de lo que está aprendiendo de una capa a otra.

De esta manera se observa que la red aprende directamente de los datos; no se tiene ninguna influencia directa sobre las funciones que se aprenden. (Figura 2).

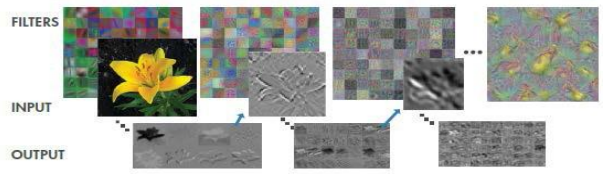

Fig 2. Entrenamiento de una CNN o una red neuronal profunda. [6].

III. CONFIGURACIÓN EN MATLAB.

El proceso de configuración de los sistemas utilizados para realizar la detección facial con el uso de redes neuronales y MATLAB consta de lo siguiente:

Debemos usar la versión 2017 (R2017b) o posterior de MATLAB, esta opción contiene las bibliotecas y la caja de herramientas de visión artificial.

Computer Vision System Toolbox™ proporciona algoritmos, funciones y aplicaciones para diseñar y simular sistemas de procesamiento de video y visión por computadora. Puede realizar la detección, extracción y comparación de características, así como la detección y el seguimiento de objetos previamente definidos en los algoritmos utilizados. Para la visión por computadora en 3-D, la caja de herramientas del sistema admite la calibración de cámara simple, estéreo y ojo de pez; visión en estéreo; reconstrucción 3-D; y procesamiento de nube de puntos 3-D. [8].

Para llevar a cabo el proceso de detección y seguimiento automático de personas en movimiento basado en cuadros de video de una cámara fija, es necesario conocer la importancia de los componentes de las aplicaciones de visión por computadora, incluido en este proceso el reconocimiento de actividad humana, monitoreo del tráfico de personas en áreas públicas o cerradas. El problema del seguimiento de objetos basado en el movimiento se puede dividir en dos partes:

- Detectar objetos en movimiento en cada fotograma.
- Asociar las detecciones correspondientes a un mismo objeto en el tiempo.

La detección de objetos en movimiento utiliza un algoritmo de resta de fondo basado en modelos de mezcla gaussianas. La manera en cómo se realiza el proceso consiste en que; las operaciones morfológicas se aplican a la máscara de primer plano resultante para eliminar el ruido. Finalmente, el análisis de blobs detecta grupos de píxeles conectados, que probablemente correspondan a objetos en movimiento. [9].

El movimiento de cada pista se estima mediante un filtro de Kalman. El filtro se utiliza para predecir la ubicación de la pista en cada cuadro y determinar la probabilidad de que cada detección se asigne a cada pista.

La función de MATLAB llamada: *initializeTracks* crea una matriz de pistas, donde cada pista es una estructura que representa un objeto en movimiento en el video. El propósito u objetivo de esta estructura o instrucción dentro de MATLAB, es mantener el estado de un objeto rastreado. El estado consta de información que se utiliza para la detección para rastrear la asignación, rastrear la terminación y mostrar. [10].

El uso de algoritmos para el aprendizaje profundo y el aprendizaje automático le permite detectar rostros, peatones y otros objetos comunes mediante detectores previamente entrenados. Con MATLAB, se puede entrenar un detector personalizado mediante el etiquetado de la verdad del terreno con marcos de capacitación como *Faster R-CNN y ACF*. También puede clasificar categorías de imágenes y realizar una segmentación semántica. [10].

La terminología y los conocimientos necesarios para el desarrollo de algoritmos y técnicas de visión aplicados bajo la técnica de *"Deep Learning"* se basan en los elementos que se muestran en la Figura 3. [11].

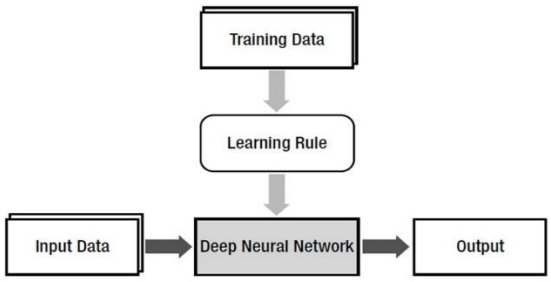

Fig 3. Elementos en los que se basa el Deep Learning. [11].

IV. METODOLOGÍA.

Este algoritmo está diseñado en Simulink®, se utiliza para detectar una cara en un cuadro de video, identificar las características faciales y rastrear estas características. El cuadro de video de salida contiene el rostro detectado y las características rastreadas. Si un rostro no es visible o se desenfoca, el sistema intenta volver a adquirir el rostro y luego realiza el seguimiento de la persona específica [12].

La figura 4. Muestra el diagrama de bloques representativo del procedimiento utilizado en el algoritmo.

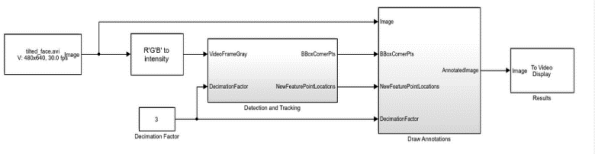

Fig 4. Muestra el diagrama de bloques representativo del procedimiento utilizado en el algoritmo.

Descripción del algoritmo.

Primero es necesario crear objetos para detectar rostros, rastrear puntos, adquirir y mostrar cuadros de video.

Es necesario crear un objeto; se llama: "*face Detector* ". En MATLAB:

faceDetector = vision.CascadeObjectDetector();

Posteriormente:

Se crea el objeto de seguimiento de puntos.

pointTracker = vision.PointTracker('MaxBidirectionalError'2);

Ahora se crea el objeto que apertura la cámara a utilizar

cam = webcam();

Para ver la lista de cámaras y sus características utilice esta línea:

webcamlist ()

Después de esta instrucción; MATLAB devolvió el nombre de dos cámaras conectadas a los puertos USB. Ver figura 5.

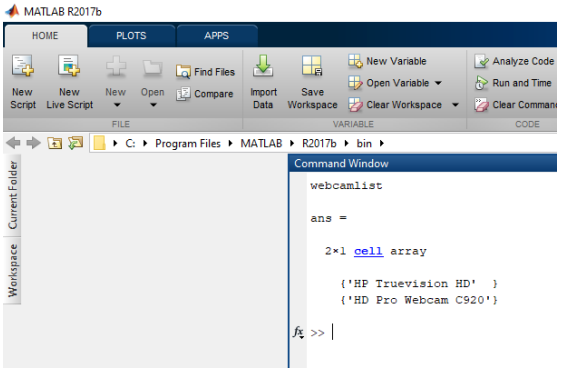

Fig 5. Muestra el nombre de dos cámaras conectadas a los puertos USB.

La figura 6 mostrada a continuación, muestra las propiedades de la cámara 1.

```
>> cam=webcam(1)

cam =

  webcam with properties:

                    Name: 'HP Truevision HD'
              Resolution: '640x480'
    AvailableResolutions: {'640x480'  '160x120'  '320x240'  '1280x720'}
                    Gain: 4
    BacklightCompensation: 0
            ExposureMode: 'auto'
            WhiteBalance: 4600
                Exposure: -6
              Saturation: 64
               Sharpness: 2
                     Hue: 0
              Brightness: 0
                   Gamma: 100
                Contrast: 0
        WhiteBalanceMode: 'auto'
```

Fig 6. Muestra las propiedades en la cámara 1.

En la figura 7 se muestra el conjunto de propiedades que reconoce el algoritmo en MATLAB para el uso de la cámara 2.

```
>> cam=webcam(2)

cam =

  webcam with properties:

                    Name: 'HD Pro Webcam C920'
              Resolution: '640x480'
    AvailableResolutions: {1×19 cell}
             FocusMode: 'auto'
                  Gain: 0
   BacklightCompensation: 0
          ExposureMode: 'auto'
                  Zoom: 100
                 Focus: 0
          WhiteBalance: 4000
                   Pan: 0
              Exposure: -5
            Saturation: 128
             Sharpness: 128
            Brightness: 128
              Contrast: 128
                  Tilt: 0
        WhiteBalanceMode: 'auto'
```

Fig 7. Muestra las propiedades en la cámara 2.

Una vez que haya elegido la cámara para probar el algoritmo, escriba el siguiente código como un archivo de script en MATLAB.

En este paso se creará el objeto que conforma el reproductor de video:

videoPlayer = vision.VideoPlayer('Position', [300 300 [frameSize(2), frameSize(1)]+30]);

Ahora es momento de crear el objeto de detección y seguimiento facial.

runLoop = true;
numPts = 0;
frameCount = 0;
while
runLoop && frameCount < 600
videoFrame = snapshot(cam);
videoFrameGray = rgb2gray(videoFrame);
frameCount = frameCount + 1;
if numPts < 10

En este punto es necesario crear la rutina del modo de detección:

bbox = faceDetector.step(videoFrameGray);

Es muy importante y necesario encontrar puntos de esquina dentro de la región detectada para limitar el tamaño del marco. Para ello se deben de usar las siguientes instrucciones:

if ~isempty(bbox)
points = detectMinEigenFeatures(videoFrameGray, 'ROI', bbox(1, :));

Hasta el momento, es importante guardar los puntos, es preciso convertir el rectángulo representado como *[x, y, w, h]* en una matriz M-por-2 de coordenadas *[x, y]* de las cuatro esquinas. Esto es necesario para poder transformar el cuadro delimitador para mostrar la orientación de la cara.

Para ello es necesario el siguiente código en MATLAB.

oldPoints = xyPoints;
bboxPoints = bbox2points(bbox(1, :));
bboxPolygon = reshape(bboxPoints', 1, []);
videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon, 'LineWidth', 3);
videoFrame = insertMarker(videoFrame, xyPoints, '+', 'Color', 'white');
end
else
[xyPoints, isFound] = step(pointTracker, videoFrameGray);
visiblePoints = xyPoints(isFound, :);
oldInliers = oldPoints(isFound, :);
numPts = size(visiblePoints, 1);
if numPts >= 10

Finalmente se debe de estimar la transformación geométrica entre los puntos antiguos y los puntos nuevos.

[xform, oldInliers, visiblePoints] = estimateGeometricTransform(...
oldInliers, visiblePoints, 'similarity', 'MaxDistance', 4);
bboxPoints = transformPointsForward(xform, bboxPoints);
bboxPolygon = reshape(bboxPoints', 1, []);
videoFrame = insertShape(videoFrame, 'Polygon', bboxPolygon, 'LineWidth', 3);
videoFrame = insertMarker(videoFrame, visiblePoints, '+', 'Color', 'white');
oldPoints = visiblePoints;
setPoints(pointTracker, oldPoints);
end
end

En este punto, se requiere mencionar que el hardware utilizado en la prueba del algoritmo es como el mostrado en la figura 8 muestra la lista de estos componentes.

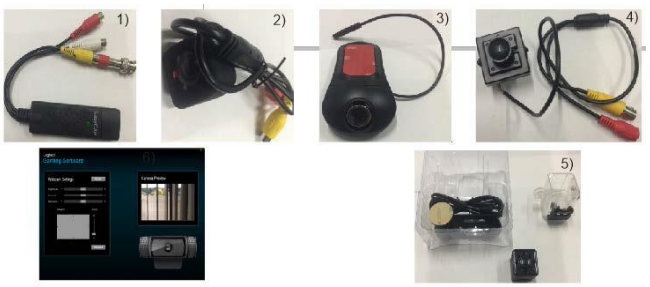

Fig. 8. Muestra los componentes de la lista utilizados en la prueba del algoritmo.

1. Tarjeta capturadora de video.
2. Cámara de reversa para uso automotriz.
3. Cámara de tablero de instrumentos de uso automotriz.
4. Micro cámara de uso convencional.
5. Micro cámara WiFi ojo de pez.

V. RESULTADOS.

La figura 9 muestra el resultado de la aplicación del algoritmo de detección, donde se puede apreciar que existe un conjunto de marcas establecidas por el programa para identificar rasgos específicos en el proceso de detección facial de la persona deseada.

Fig 9. Muestra el resultado de la aplicación del algoritmo de detección.

Se puede observar que el algoritmo indica con cruces verdes y blancas, así como un recuadro amarillo los rasgos distintivos que le permiten identificar solo el rostro de esta persona.

Cabe señalar que las pruebas se realizaron con la microcámara convencional descrita en la Figura 4; esto con el fin de demostrar que aun siendo una cámara monocular es posible aplicar el algoritmo; Se aplicaron pruebas similares con el resto de las cámaras mencionadas anteriormente en esta investigación (Cámaras: 3, 4).

VI. CONCLUSIÓN.

El aprendizaje profundo ha logrado un progreso significativo en los últimos años e incluso ha superado a los humanos en muchas tareas como la clasificación de imágenes. Por lo tanto, es atractivo e interesante reconocer que la inteligencia artificial podría llegar a acercarse o incluso superar las capacidades de los humanos de una manera más genérica.

Sin embargo, existen varios obstáculos técnicos fundamentales que deben superarse antes de que podamos construir máquinas que aprendan y piensen como personas. En particular, las redes neuronales requieren grandes cantidades de datos de entrenamiento para proporcionar resultados de alta calidad, lo que es significativamente inferior a las capacidades humanas. [13].

Algo interesante en el algoritmo es que, aunque es posible utilizar el detector de objetos en cascada en cada cuadro, es computacionalmente costoso. Esta técnica también puede fallar en la detección de la cara, como cuando la persona objetivo gira o inclina la cabeza. Esta limitación proviene del tipo de modelo de clasificación entrenado y utilizado para la detección. En este ejemplo, detecta la cara una vez y luego el algoritmo basado en KLT rastrea el rostro objetivo a través de los fotogramas de video.

La detección se realiza de nuevo solo cuando la cara ya no es visible o cuando el rastreador no puede encontrar suficientes puntos de características. Se recomienda usar métodos probabilísticos más avanzaos para predecir la siguiente posición de la cara; esta corrección probablemente reforzaría el rendimiento del algoritmo.

Hasta ahora, la primera parte del proyecto se ha completado con éxito. La siguiente fase implica el uso de un sistema DVR de uso común, cámaras en color Hik Vision con una resolución de 1080P.

Posteriormente, se creará un entorno con multitud de personas donde el algoritmo identifica a más de una persona objetivo, aprender a distinguir también entre diferentes entornos o escenarios; considerando también la aplicación de Kalman Extended Filters y nuevamente las redes CNN.

Referencias

[1] J. Schmidhuber, «Multi-column deep neural networks for image classification,» *In Proceedings of the 2012 IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3642-3649, 2012.

[2] A. L. H. A. Y. &. N. A. Y. Maas, «Rectifier nonlinearities improve neural network acoustic models.,» *In International conference on machine learning (ICML).*, pp. 120-135, 2013.

[3] S. I. Krizhevsky A, «Imagenet classificationwith deep convolutional neural networks,» *NIPS*, pp. 1-4, 2012.

[4] D. J. D. T. M. J. Girshick R, «Rich feature hierarchies for accurate object detection and semantic segmentation.,» *2014 IEEE conference on computer vision and pattern recognition (CVPR).*, pp. 580-587, 2014.

[5] J. Deng, W. Dong y S. R., «Imagenet: a large-scale hierarchical image database,» *IEEE CVPR*, 2009.

[6] A. Krizhevsky, I. Sutskever y G. Hinton, «Imagenet classification with deep convolutional neural networks,» *NIPS*, p. 1097–1105, 2012.

[7] Mathworks, «¿Qué es una red neuronal?,» 21 Octubre 2019. [En línea]. Available: https://la.mathworks.com/discovery/neural-network.html.

[8] P. Kim, MATLAB Deep Learning. With Machine Learning, Neural Networks and Artificial Intelligence, USA.: Apress, 2017.

[9] D. Bruce y L. a. T. Kanade., «An Iterative Image Registration Technique with an Application to Stereo Vision,» International Joint Conference on Artificial Intelligence, Boston M., 1981.

[10] T. &. T. K. Carlo, «Detection and Tracking of Point Features,» *Carnegie Mellon University Technical Report*, pp. 91-132, 1991.

[11] C. Andrieu, N. De Freitas y A. a. M. J. Doucet, «An introduction to MCMC for machine learning,» *Machine Learning.*, pp. 50-53, 2003.

[12] P. A. a. J. M. J. Viola, «Rapid Object Detection using a Boosted Cascade of Simple Features,» *IEEE CVPR*, pp. 112-115, 2001.

[13] Z. Kalal y K. a. M. J. Mikolajczyk, «Forward-Backward Error: Automatic Detection of Tracking Failures,» de *International Conference on Pattern Recognition*, 2010.