TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.2 , Abril-Junio, 2021, p.43- 61 ISSN 1729-3804

USO DE AI PARA LA DETECCIÓN DE LOGOS Y EL RECONOCIMIENTO FACIAL EN AMBIENTES DE DTV

Jorge A. Portal Diaz[1], Irina B. Siles Siles[2], Ania María Sánchez Pérez[3], Yusnavi Cárdenas Leiva[4]

[1-4] Universidad Central Marta Abreu de Las Villas (UCLV), Carretera a Camajuaní Km 5 ½, Las Villas, Cuba.

[1]e-mail: jportal@uclv.cu,
[2]e-mail: irinass@uclv.edu.cu,
[3]e-mail: ansanchez@uclv.cu,
[4]e-mail: ycardenas@uclv.cu

RESUMEN

En los últimos años, la inclusión de la Inteligencia Artificial, el Aprendizaje Automático, el Aprendizaje Profundo, la minería de Datos y la Ciencia de Datos han experimentado un incremento de su uso en todo tipo de sectores gracias a su adaptación a diferentes problemas o escenarios aparejado a la necesidad del procesamiento de grandes volúmenes de información. En el caso particular de la radiodifusión de TV, la ITU-R BT.2447-0 refiere que tras la inclusión de enfoques algorítmicos destacan como áreas de beneficios tecnológicos, la creación, post-producción y la distribución de programa de radiodifusión. La presente investigación, tomando como punto de partida la necesidad de obtención de metadatos a partir de archivos multimedia previamente almacenados por los productores de contenido, tiene como objetivo el uso de la extracción de características a partir de varios algoritmos de entrenamiento para resolver algunos de los problemas de la visión por computador (clasificación, localización y detección de objetos). Con tal fin, se han creado dos escenarios de estudio uno enfocado a la detección de logos (logos difundidos en las cadenas nacionales y logos de spots publicitarios nacionales) y el otro para la detección de rostros o reconocimiento facial (artistas cubanos de las artes). Se obtendrá en ambos casos para la validación de los resultados la matriz de confusión y se compararán algunos parámetros como coste computacional, métricas de evaluación, tiempo empleado, utilización de CPU o GPU, u otros para ambos escenarios.

PALABRAS CLAVES: Inteligencia Artificial, visión por computador, detección de logos, reconocimiento facial.

USE OF AI FOR THE DETECTION OF LOGOS AND FACIAL RECOGNITION IN DTV ENVIRONMENTS

ABSTRACT

In recent years, the inclusion of Artificial Intelligence, Machine Learning, Deep Learning, Data Mining and Data Science have experienced an increase in their use in all types of sectors thanks to their adaptation to different problems or scenarios coupled with the need to process large volumes of information. In the particular case of TV broadcasting, ITU-R BT.2447-0 states that after the inclusion of algorithmic approaches, the creation, post-production and distribution of broadcasting programs stand out as areas of technological benefits. The present research, starts from the need to obtain metadata from multimedia files previously stored by content producers and aims to use the extraction of characteristics from various training algorithms to solve some of the problems of the computer vision (classification, location and detection of objects). To this end, two study scenarios have been created, one focused on the detection of logos (logos broadcast on national networks and logos of national advertising spots) and the other for the detection of faces or facial recognition (Cuban artists of the arts). In both cases, the confusion matrix will be obtained for the validation of the results and some parameters such as computational cost, evaluation metrics, time spent, CPU or GPU utilization, or others will be compared for both scenarios.

INDEX TERMS: *Artificial Intelligence, computer vision, objects detection, facial recognition.*

Manuscrito recibido: 2021-01-11, aceptado: 2021-01-11
Sitio web:http://revistatelematica.cujae.edu.cu/index.php/tele

1. INTRODUCCIÓN

Según [1] en la actual revolución industrial 4.0, las tecnologías de crecimiento exponencial están protagonizadas por tecnología de sensores, inteligencia artificial (del inglés Artificial Intelligence, AI), sensores de aprendizaje automático (del inglés Machine Learning, ML), robótica, nanotecnología e impresión 3D. Hace notar de igual modo [2], que la industria inteligente comprende sistemas automatizados, distribuidos y robótica aparejados a algoritmos ML y tecnología AI para lograr un futuro rentable. El interés de la AI hoy en día, según las investigaciones de [3] y [4] es diseñar e implementar sistemas que permitan analizar grandes cantidades de datos de forma rápida y eficiente. En ambientes de producción y gestión de contenidos la UIT (Unión Internacional de Telecomunicaciones) también se ha proyectado al respecto según [5] y han profundizado en la aplicación de técnicas de AI en los archivos de televisión. En este momento, los esfuerzos actuales en curso, incluyen varias áreas de beneficios tecnológicos: optimización del flujo de trabajo, optimización del ancho de banda, creación automatizada de contenido, optimización de la selección de activos: creación de metadatos, colocación dinámica de productos y publicidad para difusión y personalización de contenidos.

El Instituto Cubano de Radio y Televisión (por sus siglas, ICRT) pretende desarrollar un proyecto de reconocimiento de logos mediante la detección de objetos en videos e imágenes. Hace algún tiempo se tiene la idea de aplicar técnicas de reconocimiento de patrones al campo de la publicidad, en concreto, al reconocimiento de identidad corporativa (logotipo, marca, imagen corporativa) en entornos audiovisuales. La idea y por ende uno de los objetivos de este trabajo consiste en analizar un evento audiovisual como spots publicitarios o programa transmitido y ser capaces mediante la tecnología de detección automática de objetos, de reconocer y cuantificar los logos de determinadas marcas que aparezcan durante la emisión del evento o para secuencias previamente almacenadas. Las estaciones de radiodifusión tienen archivos de programas de transmisión anteriores que contienen imágenes de video y otros materiales grabados. A menudo, la generación de nuevos programas se basa en la reutilización de secuencias de video almacenadas por lo que, si dichas secuencias pueden ser identificadas a partir de metadatos que indican información del contenido, la búsqueda será más fácil. Existen tecnologías de análisis de imagen y audio que incluyen reconocimiento de rostros, detección de texto de escena y detección de características de audio que permitirán generar automáticamente metadatos que describan información relevante y características de la escena según [5]. En consecuencia el segundo objetivo estaría encausado en aplicar técnicas de AI en la creación de un modelo capaz de detectar rostros de personalidades de la cultura nacional en archivos multimedia del ICRT.

Para lograr ambos objetivos, el presente trabajo aborda el entrenamiento de varios modelos de detección a partir de algoritmos que permiten extraer información del rostro para su posterior comparación y clasificación, utilizando diversas técnicas como Solo miras una vez (del inglés *You Only Look Once*, YOLO), Máquinas de Soporte Vectorial (del inglés *Support Vector Machines*, SVM), K Vecinos más Cercanos (del inglés K Nearest Neighbor, KNN) y Redes Neuronales Convolucionales (del inglés *Convolutional Neural Networks*, CNN). Por ello, se presentará el diseño de investigación empleado, los materiales utilizados, una descripción de los casos de estudios a tratar y la metodología a seguir y el análisis de los resultados.

Las contribuciones científicas fundamentales de este trabajo se establecen a partir del análisis de los resultados de cada caso estudiado enmarcados en la detección de logos y el reconocimiento facial. Como contribuciones principales se enumeran las siguientes: (1) se dota al sistema DTV de una herramienta computacional que permite la detección automática de patrones de interés mediante técnicas de AI, (2) se logró establecer una combinación de técnicas de AI que resulta novedosa para este tipo de aplicación y (3) se obtienen bases de datos anotadas personalizadas. Los resultados logrados permiten plantear la posible extensión del presente trabajo al procesamiento de otras aplicaciones que de manera similar necesiten de la detección facial eficiente y/o la detección de objetos. En las secciones siguientes se detalla el estado del arte del tema bajo análisis de manera muy suscita, se presentan los dos casos de estudio bajo análisis, el método de trabajo para cada uno de ellos y se realiza la discusión y el análisis de los resultados obtenidos.

2. ESTADO DEL ARTE

La detección de objetos se encuentra dentro del campo de la visión por computador refieren [1]–[3] y tiene como objetivo detectar todos los objetos y clases de objetos aseveran [4] y [5], además tiene un conjunto de retos según [6] y [7]. Resulta ampliamente utilizada en diversas esferas que incluyen el procesamiento

de imágenes médicas, video vigilancia, censado remoto, agricultura, detección pedestre, conducción automovilística, detección de logos, etc analizados indistintamente en [8]–[17].

En la actualidad, existen dos métodos principales de detección de objetos (1) el algoritmo de detección de objetos que combina redes neuronales convolucionales CNN y las redes neuronales convolucionales basadas en regiones (del inglés *Regional- Convolutional Neural Networks*, R-CNN) [18] y (2) detección de objetos en tiempo real (del inglés *Single Shot MultiBox Detector*, SSD) y el modelo YOLO. Asevera [17] que debido a la combinación del pensamiento de regresión de YOLO, la velocidad de detección del algoritmo SSD es la misma que YOLO, y la precisión de detección es la misma que faster R-CNN.

Tomando la línea de investigación, se encuentran en la literatura varias bases de datos de logos como [19]–[22]. Gran parte de los trabajos [23]–[27] consultados refieren ser no escalables para implementaciones de tiempo real debido a dos requerimientos: (1) etiquetado preciso para los datos de entrenamiento por clase de logos y (2) anotaciones de las cajas de anclaje o *bounding box y* [28] asevera que el mayor desafío para entrenar un detector de logotipos de propósito general radica en contar con datos de entrenamiento adecuados. En [23] y [26], por su parte se proponen métodos de detección de logos escalable denotando su mejor desempeño ante evaluaciones comparativas extensas. En [29] se utilizan las métricas de *Precision* y *Recall* como métricas de evaluación del desempeño y *Accuracy* para computar los resultados del reconocimiento. Por su parte [30] utiliza la Intersección Sobre Unión (del inglés *Intersection Over Union*, IoU), *Recall*, el Valor medio de Presición Promedio (del inglés *Mean value of Average Precision, mAP)*, y *F1-score* e implementa Yolo con la novedad de que incluye un nuevo método de inicializar la región del *bounding box*.

Por su parte, el reconocimiento de rostros es una tecnología que ha evolucionado rápidamente y que encuentra aplicaciones en verificación, identificación, fines comerciales, aplicaciones forenses, identificación criminal, control de acceso, detección de intrusos [31], seguridad en prisiones, etc. según [32]–[34]. En los trabajos de [35]–[37] se reflejan los retos para la detección de rostros tales como la oclusión, baja resolución , ruido, variación en la pose, expresiones, iluminación, longevidad, cirugías plásticas; estas mismas problemáticas [38] las resume en factores intrínsecos y extrínsecos. En ocasiones resulta muy cómodo utilizar algunos de los amplios *dataset* publicados en la *web*; para el caso de reconocimiento facial se podrían citar las bases de datos de rostros ORL, AR-fase, Gavab, BDD COMPUESTA, UMIST, FERET, CMU-PI, YALE, ESSEX, JAFFE, MS Celeb, OkCupid, etc.; pero cuando el proyecto resulta novedoso en relación a que la clasificación de imágenes para catalogación es nueva, no resulta tan trivial.

En [39], [40] y [33] se detallan de manera comparativa las diferentes técnicas de detección y reconocimiento de rostros, así como las métricas típicas asociadas. [33], hace un estudio del estado del arte de los detectores combinados con filtros y su efecto en la reducción de falsos positivos. Autores como [41] han analizado los algoritmos para la detección de rostros, [42] y [43] lo hacen para CNN, por su parte [44]–[47] para la técnica de clasificación SVM.

3. PRESENTACIÓN DEL CASO DETECCIÓN DE LOGOS HACIENDO USO DE LA AI

El presente tópico aborda el entrenamiento de un modelo de aprendizaje profundo en el marco de la red neuronal Darknet con la premisa de capturar el logotipo de la estación de TV o el logo de algunos *spots* publicitarios de producción nacional. Se decidió usar Yolo V3 [14] debido a su excelente nivel de precisión junto con el menor tiempo de entrenamiento. El presente proyecto y todos los programas complementarios necesarios durante el desarrollo fueron implementados utilizando el lenguaje de programación Python en su versión 3.6 de 64 bits. Para la detección de objetos se ha hecho uso de varias herramientas como PyTorch, TensorFlow, Keras, OpenCV y Anaconda.

Técnica de deep learning utilizada en la detección: YOLO

De acuerdo con [48] para el trabajo con YOLO existen un conjunto de prerrequisitos que van desde comprender cómo funcionan las redes neuronales convolucionales, pero que además toma en consideración el dominio cognitivo de Bloques residuales, conexiones de omisión y *Upsampling*. Se necesita un bagaje adicional sobre la detección de objetos [49], la regresión del cuadro delimitador [50], IoU y la supresión no

máxima [51]. Se afirma en [52] qué es una red neuronal completamente convolucional (del inglés *Fully Convolutional Networks*, FCN) y en consecuencia un detector de objetos que utiliza las características aprendidas por una red neuronal convolucional profunda, con sus respectivas capas convolucionales, para detectar un objeto. Sobre cuestiones prácticas, se asume por [53] que uno de los grandes problemas es que si se quiere procesar las imágenes en lotes en paralelo por la GPU, se precisa de que todas las imágenes tengan altura y ancho fijos. Las ventajas que posee este algoritmo se enuncian en [54].

En modelo YOLO está pre-entrenado para el *dataset* de CoCo (del inglés *Common Objects in Context*) [55], por lo que puede ser utilizado directamente para hacer predicciones sobre unas categorías determinadas. Este *dataset* cuenta con un total de 300 000 imágenes etiquetadas que contienen objetos de 80 clases diferentes, divididas en varios grupos. En ocasiones y esta investigación es un ejemplo de ello, se puede dar el caso en el que los objetos que se pretenden detectar simplemente no se encuentran en los modelos preentrenados que ofertan estos *framework* de programación. Para esta situación se hace imprescindible crear (u obtener) un propio set de datos y ejecutarlo en consecuencia para el entrenamiento de la red.

La construcción de un modelo de detección de objetos generalmente requiere una gran cantidad de datos de entrenamiento etiquetados recogidos de etiquetado manual extenso según [56] y [57]. La mayoría de los estudios existentes sobre la detección de logos se limita a escalas pequeñas, tanto en el número de imágenes de logotipo y clases de logotipo [58], [59], en gran parte debido a los altos costos en la construcción de conjuntos de datos de logotipos a gran escala. No es trivial recopilar automáticamente los logotipos a gran escala para datos de entrenamiento que cubren una gran cantidad de diferentes clases de logotipos. Si bien la minería de datos *web* puede ser una solución potencial como se muestra en otros problemas de reconocimiento [60]–[62], es difícil obtener anotaciones de logotipo precisas ya que no siempre la anotación del cuadro delimitador está disponible en imágenes *web* típicas y sus metadatos.

Entrenamiento de un nuevo modelo

En esta sección se explica cómo se cosntruyó un modelo de detección de objetos en imágenes, específicamente se detectaron logotipos de televisión difundidos en las cadenas nacionales y provinciales o logos de *spots* publicitarios encontrados en las redes sociales. Esto se realizó siguiendo los pasos que se muestran en el esquema de la Figura 1. El diagrama se elaboró a partir de analizar los trabajos de [63]–[66]. Destacar que al flujo de trabajo elaborado parte de imágenes que constituyen un conjunto de datos sin etiquetar y que en este estudio puntual tras no contar con la suficiente cantidad de imágenes se llevó a cabo la técnica del aumento de datos, etapa que es defendida por [67] y [68] con las clásicas técnicas que no llegan a ser tan novedosas como las planteadas por [69]. La versión mostrada constituye la adecuación de la idea de [70] que usa un esquema subdividido en 3 fases (anotación, entrenamiento e inferencia). La finalidad de cada una de estas etapas se detalla a continuación.

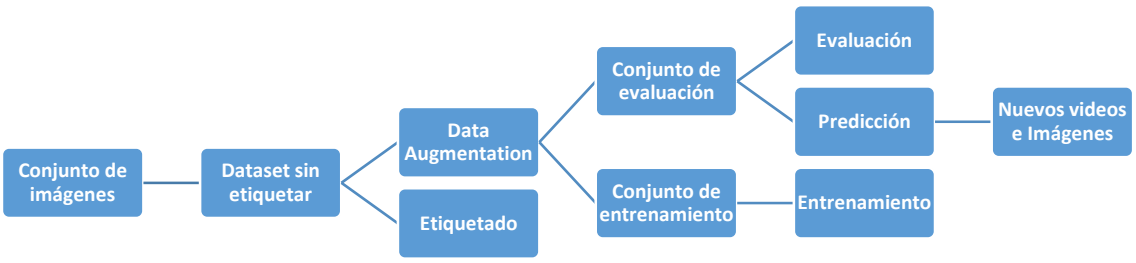


Figura 1. Pasos necesarios para entrenar un nuevo modelo.

Obtención de un dataset de logos

Se obtuvo un *dataset* preliminar del proyecto FlickrLogos [71] de la universidad de Augsburg. El conjunto de datos viene en dos versiones: FlickrLogos-32 diseñado para la detección y evaluación de objetos de varias clases y FlickrLogos-47 que ya incluye todas las anotaciones para la tarea de detección y reconocimiento de objetos. Esta investigación al igual que la de [10] propone que los experimentos se lleven a cabo en la base de datos FlickrLogos-32 con un conjunto de datos que ellos nombraron Logos-32plus ampliado y por nuestra parte FlickrLogos-32 UCLV.

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.2 , Abril-Junio, 2021, p.43- 61 ISSN 1729-3804

Data augmentation

Una práctica común dentro de la clasificación de imágenes mediante redes neuronales es el aumento de los datos presentes en el *dataset*. Especialmente útil donde la proporción de clases no está balanceada y existen numerosas categorías con una única muestra. Para aumentar el número de datos disponibles para el entrenamiento se realizan una serie de procesados sobre una imagen original y así generar una serie de imágenes derivadas. Entre las posibles modificaciones existentes se encuentran: traslación, rotación, filtros, voltear, escalar, entre otros [13], esto conlleva a aumentar la variabilidad de las imagines y hacer que la red sea capaz de recibir datos variados y pueda clasificar con mayor exactitud. Si una imagen es modificada ligeramente es percibida por la red como una imagen completamente distinta perteneciente a la misma clase. Esto disminuye las probabilidades de que la red se centre en orientaciones o posiciones mientras mantiene la relevancia de las características deseadas.

Anotación del dataset

Toda una vez que se han obtenido las imágenes se hace necesario implementar la anotación de las mismas o la anotación del *dataset*; esto quiere decir que a cada una de las imágenes se le ha de proporcionar de manera manual una etiqueta. Estas etiquetas han de indicar además de la categoría a la que pertenecen cada uno de los objetos y la posición de cada objeto dentro de la imagen en cuestión, a lo que se llama anotación. Refiere [13] que para cada modelo de detección se utiliza un formato de anotación específico; para el caso de YOLO v3 se requiere un archivo .txt para cada una de las imágenes con una línea para cada objeto a detectar, con la siguiente forma <*object-class*><*x*><*y*><*width*><*heigth*>. Donde <*object-class*> es la clase del objeto, <*x*> e <*y*> son las coordenadas del centro del objeto y <*width*> y <*height*> son el ancho y el alto del rectángulo que contiene al objeto. Este problema de investigación únicamente requiere la detección de una clase de objetos (logos) pero que se pueden presentar en formatos disímiles, se podría valorar el uso de alguna herramienta de anotación como BBox-Label-Tool, Lionbridge AI, LabelImg o YOLO mark. Por su parte LabelImg [72] aseveran que se pueden realizar anotaciones en forma de *bounding boxes* de una forma ágil y eficaz, en formatos YOLO (.txt) y Pascal VOC (.xml) y fue la seleccionada.

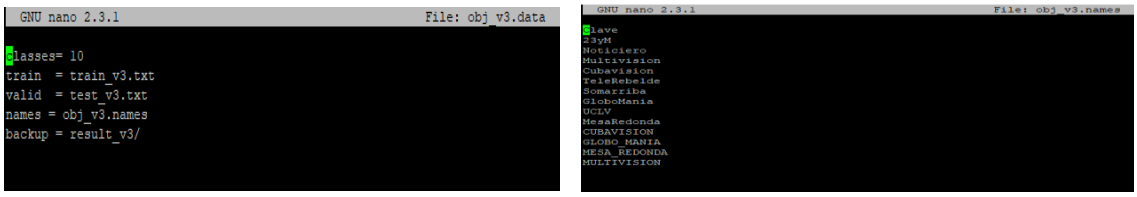
Antes de comenzar con el proceso de anotación se cuenta típicamente tamaños de imágenes con una resolución de 1366x768, siendo el tamaño de los logos de 654x490 en el 55% de los casos para cuando el logo se encuentra en el centro y de 235x160 en el 71% para cuando se encuentran en un extremo. Entiéndase, que de acuerdo al esquema de anotación de YOLO, para el caso de la Figura 2 (a) existen dos líneas en el archivo .txt porque se da el caso de que en un mismo fotograma aparecen dos muestras de una misma clase de objeto (clase 15 nombrada Mesa Redonda) y para el caso de la Figura 2 (b), existe una única clase de objeto (clase 3 nombrada Globo Manía). De manera ilustrativa y para una mejor comprensión de los lectores, el uso de anotaciones en forma de *bounding boxes* se toma como referencia la Fig 2 (b) donde los números 0.494483 y 0.372208 representan las coordenadas del centro del objeto a etiquetar y 0.539310 y 0.163772 son el ancho y el alto del rectángulo contenedor del logo.

(a) (b)

Figura 2. Proceso de anotación con LabelImg con más de un logo en la imagen con su archivo .txt (a) y para un logo con su archivo .txt para (b).

División del conjunto de datos

A decir de [73] un sobreentrenamiento o *overfitting* podría generar un sobreajuste del modelo y una mala generalización; esto se manifiesta de forma tal que el modelo no será capaz de encontrar patrones que no se encontrasen anteriormente en el conjunto de datos empleados para el entrenamiento. Para mitigar esta situación el conjunto principal se divide en dos subconjuntos o partes. En YOLO este proceso de creación de los subconjuntos es manual ya que son los propios programadores de la aplicación los que dividen el *dataset* en dos o tres partes. Esto conlleva además la aparición de dos conjuntos independientes en las que se almacenan las imágenes de entrenamiento en una carpeta y las de evaluación/test en otra diferente, en este caso la diferenciación se realiza mediante .txt que se define el *path* de cada una de las imágenes implicadas en este proceso. Se define en el fichero obj.data (véase la Figura 3 a) donde se dividen los conjuntos, como se puede observar en la imagen, se definen además la cantidad de clases, el fichero donde se encuentra el nombre de la etiqueta correspondiente en el .txt creado en el proceso de anotación (fichero obj. names, véase la Figura 3 b) y la carpeta donde se guarda el modelo entrenado.



(a) (b)

Figura 3. Fichero obj. Data(a) y Fichero obj. Names (b)

Entrenamiento

Previa realización de la secuencia de pasos descritos previamente, encausados en el procesamiento de las imágenes (organización en carpetas correspondientes y anotaciones de las mismas) se ha de entrenar un modelo capaz de detectar y clasificar los objetos (logos) dentro de un fotograma. [13] afirma que lograr el entrenamiento de una red neuronal artificial es un proceso en el que se desea encontrar una configuración óptima de pesos, de tal manera que la red pueda extraer información útil (o patrones) de los datos que se le pasan por entrada, para luego poder generalizar. Los pesos - *weights* son coeficientes que se van adaptando dentro de la red a medida que esta va aprendiendo a partir de los ejemplos de entrenamiento, en el caso de la red utilizada se guarda un *archivo.weights* cada 200 iteraciones hasta llegar a 20000.

En el caso de YOLO v3 y según [74] existen dos opciones para llevar a cabo el entrenamiento, YOLO se puede entrenar desde cero o se pueden utilizar unos pesos previamente entrenados (provenientes del modelo darknet53 pero no estén guiado a la tarea en específico que se desea realizar, por lo que se debe re-entrenar con los ajustes de cantidad de clases que se deben clasificar para las capas convolucionales). En cualquier caso, se hace imprescindible pasar como entrada las imágenes anotadas. Es preferible usar la segunda opción de entrenamiento ya que se parte de una red que ya ha sido previamente entrenada y que no le costará demasiado encontrar la configuración óptima de pesos. Esto aminora tiempo y esfuerzo. Para lograr configurar el entrenamiento de la red YOLO se utiliza como base el fichero yolo-obj.cfg que contiene además de la estructura de la red YOLO otros parámetros que serán necesarios modificar.

Creación de las capas de la arquitectura de red

Trabajando secuencialmente lo primero sería crear un directorio donde viva o radique el código del detector. Luego, se ha de crear un archivo darknet.py ya que Darknet es el nombre de la arquitectura subyacente de YOLO. Este archivo contendrá el código que crea la red YOLO. Hay además que complementar con un archivo llamado util.py que contendrá el código para varias funciones auxiliares. Ambos archivos serán guardados en la carpeta del detector. Se utilizó ***git*** para realizar un seguimiento de los cambios efectuados, en este caso el GitLab de la UCLV **https://git.uclv.edu.cu**. La siguiente instruccion define el comienzo del nuevo entrenamiento:

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.2 , Abril-Junio, 2021, p.43- 61 ISSN 1729-3804

./darknet detector train obj.data yolov3.cfg cfg/darknet19_448.conv.23

Una vez terminado el proceso de entrenamiento se obtiene varios *archivos .weights* con los pesos (cada cierto número de iteraciones se guarda un *archivo.weights*) que van a permitir a la red detectar y clasificar los objetos de interés. De estos pesos se debe elegir el mejor de ellos. Con dichos archivos, YOLO ya podrá detectar y clasificar las clases que se le han indicado previamente. Para saber si la red ha sido entrenada correctamente hay que pasar a evaluarla, véase la Figura 4 y nótese que dicho entrenamiento llevó seis días (del 11 al 17 de agosto/2020) para completar los 20000 weigths.

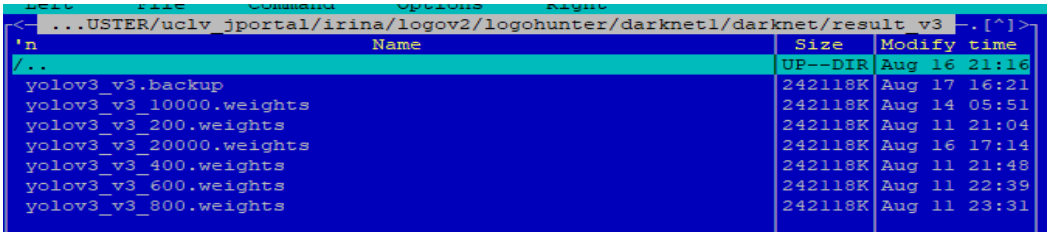


Figura 4. Archivo .weights

Evaluación

En esta etapa se pone a prueba la información o conocimiento que la red ha obtenido del entrenamiento en el paso anterior. Se evalúa si la red es precisa en sus predicciones y si no se está conforme con su rendimiento, se puede volver a la etapa anterior y continuar entrenando la red cambiando algunos de los parámetros hasta lograr un rendimiento y eficacia aceptable. El proceso de evaluación consiste en pasar las imágenes de test por la red (es muy importante que no se mezclen los conjuntos), para obtener los resultados de detección obtenidos por la red y compararlos con los de la realidad. La finalidad de este proceso permitirá extraer información acerca del comportamiento del modelo y datos relacionados con el rendimiento (como puede ser la tasa de aciertos o de fallos) que ayudarán a decidir si el modelo es adecuado o no al problema. En el caso de YOLO, durante la etapa de entrenamiento se obtienen varios *ficheros .weights* y hay que elegir el que mejor funcione. Para seleccionar uno de estos ficheros hay que fijarse en el que obtenga la IoU más altas.

Predicción

Una vez seleccionado el archivo con los mejores pesos, el siguiente paso será poner en producción el modelo y empezar a utilizarlo en imágenes que no han sido utilizadas ni para entrenar ni para realizar los tests. Hay distintas maneras de realizar dicha predicción como se muestra a continuación. Para una única imagen se usa el siguiente comando: **./darknet detect cfg/yolo-obj.cfg yolov3.weights data/UCLV.jpeg**

Por defecto, YOLO no muestra las detecciones, pero sí que las guarda en un fichero llamado predictions.png y desde aquí se pueden ver los objetos detectados como por ejemplo en la Figura 5.

Figura 5. Detección eficiente de tras correcto entrenamiento para algunos de los logos trabajados.

4. PRESENTACIÓN DEL CASO INCLUSIÓN DE TÉCNICA DE AI PARA LA CATALOGACIÓN Y DETECCIÓN DE ROSTROS DE PERSONALIDADES DE LA CULTURA CUBANA

El reconocimiento facial es una aplicación informática capaz de detectar, rastrear, identificar o verificar rostros humanos a partir de una imagen o video capturado con una cámara digital. Este proyecto de investigación utiliza detección de rostros en imágenes, así como su posición, particularmente de reconocidas personalidades de la cultura cubana que trascienden en la esfera internacional. Se pretende elaborar varios modelos de detección que permitan determinar y evaluar algunos parámetros como la precisión, sensibilidad, exactitud, especificidad, entre otros. El lenguaje de programación que se va a utilizar a lo largo del desarrollo del proyecto es Python 3.6, el entorno de desarrollo utilizado ha sido Jupyter 1.0.0, disponible bajo la distribución multiplataforma Anaconda, versión 1.9.6 [75]. Para la implementación del sistema se han usado una serie de librerías que han facilitado el desarrollo gracias a una serie de funciones o clases que llevan implementados los algoritmos explicados durante este trabajo. Las principales librerías que se han utilizado son: NumPy (versión 1.16.5), OpenCV (versión 4.30), Face Recognition (versión 1.2.3), Scikit-Learn (versión 0.21.3), TensorFlow (versión 2.3.0), Pytorch (versión 1.1.0), OpenFace (2.2.0) y Dlib (versión 19.20.0).

Elaboración de un modelo de trabajo

Recordar que unas de las finalidades de esta investigación pueden ser la búsqueda en programas de archivo que son utilizados por otros espacios de la Televisión Cubana o la recolección de momentos en escena de alguna personalidad para la confección de crónicas o dramas biográficos; situaciones identificadas por el autor pero que no sesgan la propuesta únicamente a este escenario y si aplican a otras opciones de video vigilancia, de control de registro de entrada de empleados, de autenticación y admisibilidad de la evidencia electrónica obtenida de perfiles de redes sociales, etc.

Generalmente, un sistema de reconocimiento facial se divide en cuatro fases: detección, preprocesado, extracción de características, comparación y clasificación, de acuerdo con [128]. A decir de otros investigadores [76]. [108], [77] y [130] se ha de incluir una fase de procesado de la imagen antes de la detección, mientras otras investigaciones como [109] [129] y [110], resumen las fases en 3 excluyendo la fase de preprocesamiento ya que alegan que en la propia fase de detección la imagen ya se encuentra preprocesada y lista para la extracción de características. Tomando en consideración estas ideas, se elaboró un esquema de trabajo de acuerdo a la Figura 6 donde se parte de la revisión bibliográfica de temas asociados a la detección eficiente de rostros. Seguidamente, a la par que se seleccionan los posibles métodos de detección a implementar de acuerdo con [46] se va trabajando en la obtención del conjunto de datos y la segmentación de los mismos en train y test. Posteriormente, y sobre los datos de entrenamiento, se confecciona una etapa para la implementación de algoritmos para la deteccion de rostros, le sigue otra para la extracción de características y luego la última fase de comparación y clasificación. El experimento concluye con la evaluación de los resultados obtenidos tras el análisis de algunas métricas. Los tópicos siguientes detallan cada una de estas etapas.

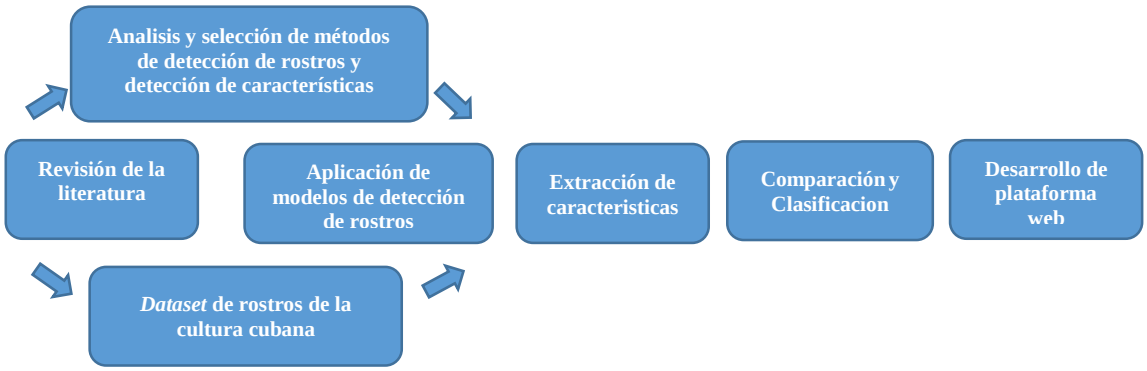


Figura 6. Metodología de la construcción del proyecto.

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.2 , Abril-Junio, 2021, p.43- 61 ISSN 1729-3804

Construcción de un *Dataset* propio para el reconocimiento facial

En varias investigaciones [78]–[81] se aborda de lo monótono y demorado que puede resultar la creación de un *dataset*, sobre todo si se trata de lograr que sea extenso en cuanto a la cantidad de clases a clasificar y vasto en relación a la cantidad de imágenes para cada una de las clases. Para construir "rápidamente" un conjunto de datos de imágenes de aprendizaje profundo, se utilizó la API de búsqueda de imágenes Bing de Microsoft accesible desde [82] y [83]. Esta API forma parte de los servicios cognitivos de Microsoft que se utilizan para llevar la inteligencia artificial a la visión, al habla, al texto, a las aplicaciones y al software. Con dicha herramienta la descarga de aproximadamente 700 imágenes no rebasó las dos horas, trabajo que hubiese llevado al menos ocho horas. Esclarecer además que para el caso del reconocimiento facial no se precisa, como para el caso de la detección de objetos, un número tan extenso de imágenes para cada clase; esto se debe al hecho de que cada rostro tiene características muy inherentes y peculiares que lo hacen único tal cual las huellas dactilares, las orejas, etc.

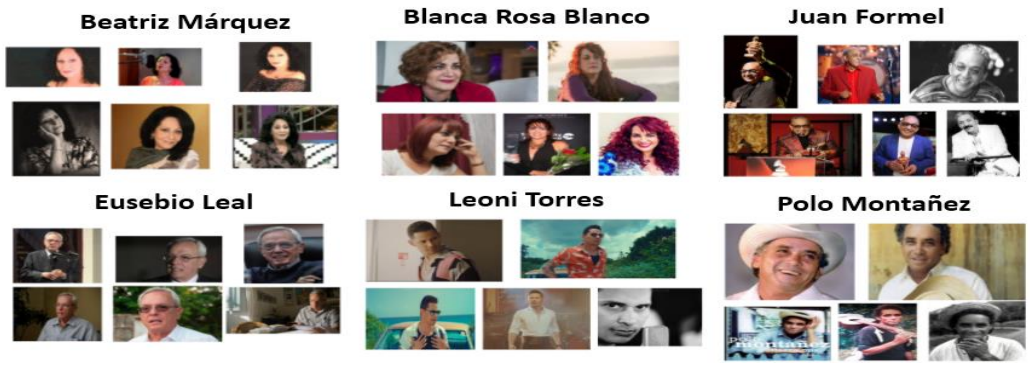


Figura 7. Ejemplo del *dataset* obtenido para la detección facial a partir de la API Bing Image Search.

Procesamiento de reconocimiento facial

En pocas palabras, un sistema de reconocimiento facial ha de extraer características de la cara o el rostro de una persona para una imagen de entrada y las ha de comparar con las características fisionómicas de las caras etiquetadas en una base de datos. La comparación se basa en una métrica de similitud de características y la etiqueta de la entrada de la base de datos más similar se usa para etiquetar la imagen de entrada. Si el valor de similitud está por debajo de un cierto umbral, la imagen de entrada se etiqueta como desconocida. La figura a continuación muestra para (a) una imagen detectada de manera idónea y otra (b) en la que fue capaz de detectar el rostro, pero no identificó la persona y la etiquetó como desconocida.

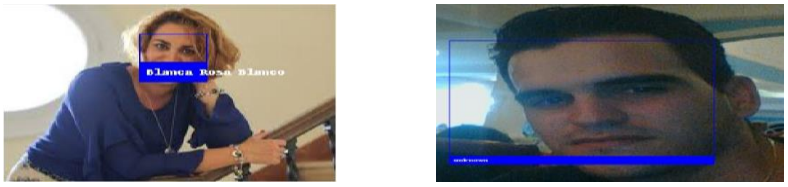
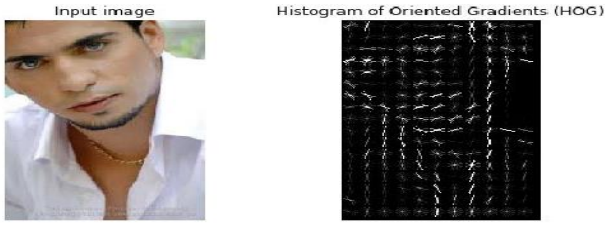


Figura 8. Imágenes reconocidas de forma eficiente (a) y fallida (b).

Detección

La detección facial es el proceso mediante el cual el sistema localiza la posición de los rostros de personas en una imagen o fotograma. La detección de rostros se generalizó a principios de la década de 2000 cuando Paul Viola y Michael Jones inventaron una forma de detectar rostros que era lo suficientemente rápida como para funcionar con cámaras baratas [84]. Sin embargo, ahora existen soluciones mucho más

confiables como el método basado en Redes Neuronales Convolucionales [85] y el Método de los histogramas de gradientes orientados, o simplemente HOG (del inglés, *Histogram of oriented gradients*) [86] para abreviar. Patentado en el 2005, HOG será el que se utilice para implementar el sistema de reconocimiento facial de este proyecto ya que ha sido demostrada su eficiencia por [87]–[90] y [14]. En la figura siguiente se puede observar una imagen original y, a la derecha, el resultado de calcular la imagen HOG siguiendo el método descrito por [76].



a) b)

Figura 9. Obtención de la imagen HOG para una de las imágenes: imagen original (a) y (b) imagen HOG.

Para el caso de las redes CNN se trabaja con redes pre-entrenadas y el proceso no es exactamente el mismo. No obstante a continuación se muestran algunos de los resultados obtenidos para la detección de rostro para Liuba María Hevia (Figura 10).

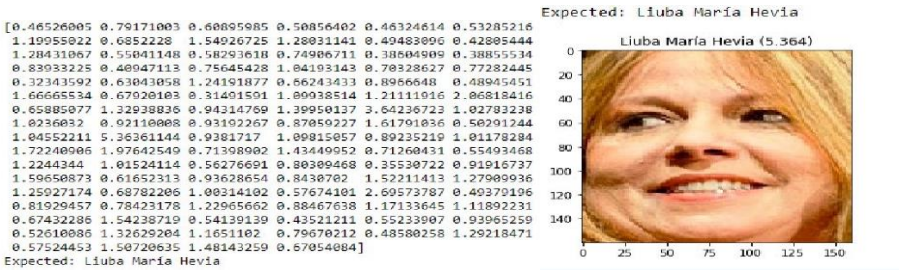


Figura 10. Detección de rostro usando *Deep Learning.*

Preprocesamiento

En esta etapa hay que lidiar con un problema, y es que el rostro de una misma persona girado en dos posiciones distintas pudiera ser, para la computadora, dos personas diferentes. Para ajustar la imagen se hace uso de un algoritmo llamado estimación de punto de referencia que ha sido implementado en la librería *Face Recognition*. Concretamente, el algoritmo se basa en el trabajo de V. Kazemi y J. Sullivan [91], el cual consiste en estimar una serie de *p* puntos de referencia de la cara de una forma eficiente, haciendo uso de una cascada de regresores. La idea básica es que se crean 68 puntos específicos (llamados *face landmarks*) que existen en cada cara: la parte superior del mentón, el borde exterior de cada ojo, el borde interior de cada ceja, etc. La librería dlib aporta el estimador ya entrenado para encontrar los puntos faciales según [92].

En Figura 11 se muestran los arreglos obtenidos para algunas imágenes de prueba, nótese que en los tres c asos se obtienen los *landmark*

Figura 11. Obtención de los *landmark* para Osdalgia, Carlos Varela , Alicia Alonso y Haila.

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.2 , Abril-Junio, 2021, p.43- 61 ISSN 1729-3804

Extracción de características

Según [14] la extracción de características consiste en la aplicación de algoritmos a imágenes digitales para reducir la redundancia y las irrelevancias que esta presenta. El proceso de formación funciona observando tres imágenes faciales a la vez se inializa cargando una imagen de la cara de entrenamiento de una persona conocida, luego se carga otra foto de la misma persona conocida y por último se carga una foto de una persona totalmente diferente. Luego, el algoritmo analiza las mediciones que está generando actualmente para cada una de esas tres imágenes. Seguidamente, ajusta ligeramente la red neuronal para asegurarse de que las medidas que genera para el fotograma #1 y el fotograma #2 estén un poco más cerca mientras se asegura de que las medidas para los fotogramas # 2 y #3 estén un poco más separadas [93].

Después de repetir este paso millones de veces para millones de imágenes de miles de personas diferentes, la red neuronal aprende a generar 128 mediciones de manera confiable para cada persona. Diez imágenes diferentes de la misma persona deberían dar aproximadamente las mismas medidas, recordar que la media de imágenes usadas en este proyecto fue de 7.6 imágenes por clase como promedio. Pero una vez que la red ha sido entrenada, puede generar medidas para cualquier rostro, incluso para aquellos que nunca antes había visto. Por lo tanto, este paso solo debe realizarse una vez a decir de [93]. Es por eso que OpenFace resulta una ventaja pues provee varias redes capacitadas que se pueden usar directamente, tan solo se deben ejecutar las imágenes faciales a través de su red previamente entrenada para obtener las 128 medidas para cada rostro.

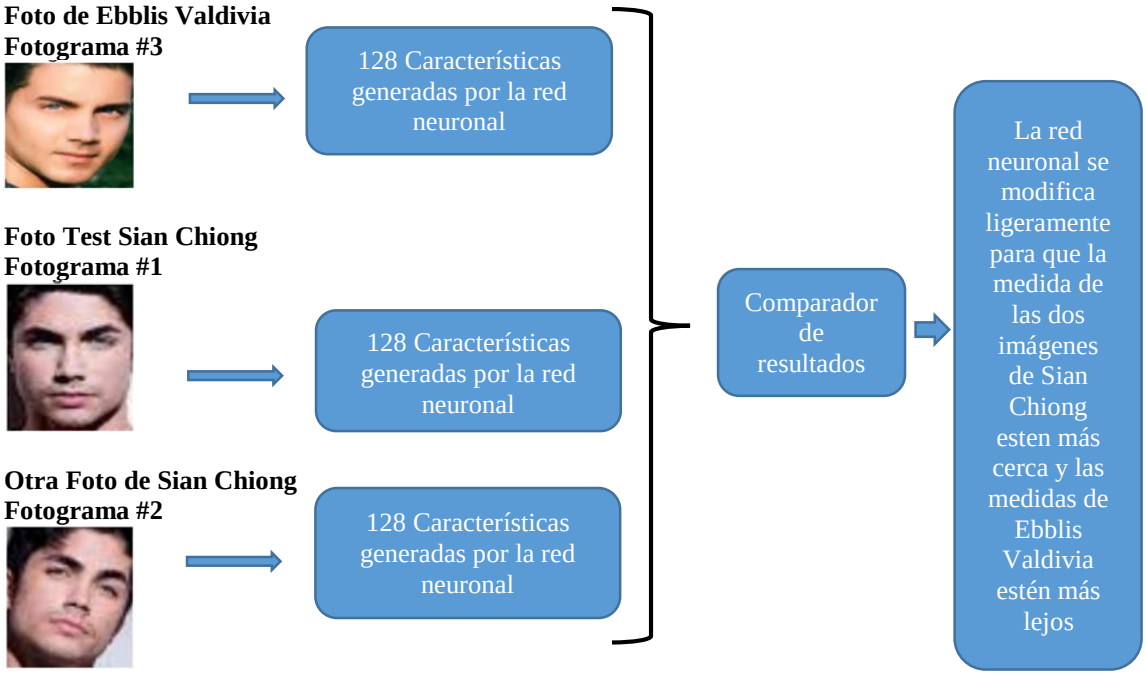


Figura 12. Diagrama ilustrativo para el uso de un único paso de entrenamiento de "triplete"

Comparación y Clasificación

A lo largo de todo el proceso se han ido utilizando diferentes técnicas y algoritmos para obtener, finalmente, una representación fiel de la cara que aparece en una imagen dada. La última fase de un sistema de reconocimiento facial consistirá en la evaluación de dicha representación, de modo que pueda ser comparada con la base de datos y poder llegar a lograr una clasificación, determinando si se corresponde con alguna de la base de datos o si, por el contrario, se trata de una cara desconocida [76]. Según los métodos que hayan sido utilizados en los procesos anteriores, será conveniente el uso de unas técnicas de

comparación u otras. En este proyecto se implementaron las siguientes técnicas de comparación y clasificación de datos: KNN y SVM. Algunos de los resultados se muestran en la Figura 13.

Figura 13. Rostros detectados de forma correcta.

5. RESULTADOS Y DISCUSION PARA AMBOS CASO DE ESTUDIO

Resultados para el caso Detección de Logos haciendo uso de la AI

Para evaluar el proceso de la detección de objetos se usan varias métricas que caracterizan la precisión del detector en un conjunto de datos particular. Una de estas métricas es la conocida como IoU que permite evaluar cuán similar es el cuadro delimitador predicho en relación al cuadro delimitador veraz o real fundamental. La Figura 14 muestra una pequeña porción de los resultados obtenidos para una época dada donde en función de una región se obtienen valores de IoU. Estos valores de IoU como se muestran están asociados a regiones y tienen una tendencia a mejorar a medida que aumenta el número de iteraciones.

```
Region 82 Avg IOU: 0.469976, Class: 0.321775, Obj: 0.034651, No Obj: 0.017333, .5R: 0.5000, .75: 0.000, count: 2
Region 94 Avg IOU: 0.104179, Class: 0.329022, Obj: 0.001767, No Obj: 0.004969, .5R: 0.0000, .75: 0.000, count: 1
Region 106 Avg IOU: 0.431184, Class: 0.752649, Obj: 0.000583, No Obj: 0.00232, .5R: 0.0000, .75: 0.000, count: 1
Region 82 Avg IOU: 0.275605, Class: 0.562945, Obj: 0.092825, No Obj: 0.017333, .5R: 0.0000, .75: 0.000, count: 2
```

Figura 14. Resultados obtenidos para una época en función de valores de IoU.

Sin embargo, la IoU no tiene en cuenta la clase y se define para un solo objeto, mientras que cuando se evalúa un modelo de detección se quiere hacerlo con respecto a un conjunto de *test* y teniendo en cuenta la clase de los objetos detectados, por lo que la métrica de IoU no se puede utilizar directamente. Para resolver este problema se define la métrica mAP, la misma se puede hallar calculando la precisión promedio por separado para cada clase, luego el promedio sobre la clase. Una detección se considera un verdadero positivo solo si el mAP es superior a 0,5.

La Tabla 1 muestra los resultados obtenidos para algunas de las variables discutidas para lo que se llamó Data Split 1 y Data Split 2. Entiéndase que Data Split 1 fue una primera corrida donde los fotogramas para las carpetas *Train* y *Test* se escogieron aleatoriamente, es decir las imágenes que estaban contenidas en cada una de las carpetas podían o no tener diferencias cromáticas de los logos, variaciones en la resolución más o menos bruscas, ubicaciones en cualquier parte de la imagen, etc. Para el Data Split 2 se sopesaron algunas de las deficiencias detectadas tras el entrenamiento del Data Split 1 y las imágenes contenidas en las carpetas *Test* y *Train* estaba "equilibradas" o eran más representativas.

Tras terminado el entrenamiento para el *weigths* 20000 se analizan los resultados obtenidos y se tabulan para el 10% de las imágenes de las 10 clases que formaban parte de test. Para una media de 99.3% se detectan de manera eficiente los fotogramas para el Data Split 2. Las clases que obtienen mejor *score* son Mesa Redonda, Noticiero, 23 y M, Eventos Somarriba, Eventos Globo Manía y los de la UCLV; véase respectivamente imagnes asociadas a la Figura 5.

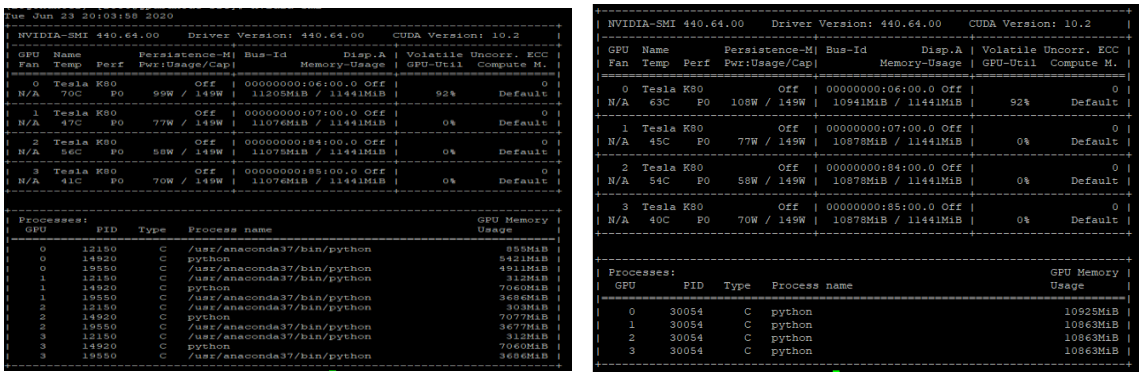
TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.2 , Abril-Junio, 2021, p.43- 61 ISSN 1729-3804

Tabla 1 Resultados tabulados para Data Split 1 y 2 para las métricas *Presicion, Recall e IoU* en función de las iteraciones.

	Mejor mAP	Iteracion	*Presicion*	*Recall*	*Avg IoU*
Data Split 1	0,9865	1000	0,9455	0,9831	0,9322
	0,9883	5000	0,9554	0,9851	0,9455
	0,9958	10000	0,9688	0,9899	0,9886
	0,9961	15000	0,9699	0,9899	0,9892
	0,9963	20000	**0,9899**	**0,9901**	**0,9919**
Data Split 2	0,9921	1000	0,98	0,9911	0,9881
	0,9881	5000	0,9912	0,9921	0,9885
	0,9879	10000	0,9931	0,9954	0,9923
	0,9955	15000	0,9958	0,9971	0,9966
	0,9985	20000	**0,998**	**0,9989**	**0,9988**

En el entrenamiento de logos el empleo de GPU ayudó en el proceso de alcanzar y ajustar con mayor rapidez los experimentos realizados, en este caso se emplearon 2 K80 en paralelo en un server R730 con 148 GB de Memoria RAM . Se realizaron ajustes con el objetivo de ocupar todos los hilos de procesamiento de la CPU como de la GPU. En el caso del uso de la CPU el proceso de escalamiento en la memoria RAM no había problema ya que el servidor cuenta con 148 GB de RAM para CPU, mientras que el uso de la GPU se realizaron ajustes en cuanto el número de imágenes en el *batch* ya que un número máximo a 64 imágenes provocaba colapso en la memoria RAM de la tarjeta GPU y hacía que el proceso de entrenamiento se detuviera, en algunos casos el proceso iniciaba perfectamente pero a medida que se realizaban iteraciones aumentaba la memoria consumida hasta el tope máximo y el proceso caía de igual manera. En este caso se obtuvo por memoria compartida y al hacer uso solamente de 2 GPU de las cuatro disponibles.

Los tiempos de ejecución usando GPU eran casi cuatro veces más rápidos que empleando CPU, en este caso se midió exclusivamente para la demostración el tiempo consumido para realizar las primeras iteraciones. Se puede observar que, en GPU solamente tardó 17 segundos aproximadamente en realizar una iteración, además de realizar saltos en las iteraciones, en la secuencia mostrada se puede observar perfectamente el salto de tres iteraciones a ocho iteraciones con el mismo tiempo, mientras que con el uso de CPU se realiza iteración a iteración con un tiempo de computo de aproximadamente 39 segundos, aproximando por exceso. En esta comparación se emplearon las mismas configuraciones para el entrenamiento, con el objetivo de tener una comparación justa.



(a) (b)

Figura 15. Uso de memoria de la GPU para algunas de las pruebas realizadas.

```
2: 23333.488281, 23240.765625 avg, 0.000000 rate, 34.995789 seconds, 128 images
2: 23333.488281, 23240.765625 avg, 0.000000 rate, 34.995789 seconds, 128 images
6: 25779.427734, 25781.718750 avg, 0.000000 rate, 38.676485 seconds, 384 images
7: 25759.671875, 25779.513672 avg, 0.000000 rate, 38.715423 seconds, 448 images
8: 25722.253906, 25773.787109 avg, 0.000000 rate, 38.647577 seconds, 512 images
9: 25856.423828, 25782.050781 avg, 0.000000 rate, 38.878467 seconds, 576 images
10: 25817.429688, 25785.587891 avg, 0.000000 rate, 38.564408 seconds, 640 images
```

Figura 16. Resumen del uso de CPU para 64 imágenes en batch.

```
1: 10495.853516, 10495.853516 avg, 0.000000 rate, 17.481643 seconds, 128 images
2: 10504.679688, 10496.736328 avg, 0.000000 rate, 17.064994 seconds, 256 images
3: 10512.056641, 10498.268555 avg, 0.000000 rate, 17.012497 seconds, 384 images
8: 10486.895508, 10497.130859 avg, 0.000000 rate, 18.025011 seconds, 1024 images
9: 10496.828125, 10497.100586 avg, 0.000000 rate, 17.105950 seconds, 1152 images
10: 10520.263672, 10499.416992 avg, 0.000000 rate, 17.658073 seconds, 1280 images
11: 10507.691406, 10500.244141 avg, 0.000000 rate, 17.269382 seconds, 1408 images
16: 10506.708984, 10500.890625 avg, 0.000000 rate, 17.696130 seconds, 2048 images
17: 10510.937500, 10501.895508 avg, 0.000000 rate, 16.946116 seconds, 2176 images
18: 10525.066406, 10504.212891 avg, 0.000000 rate, 17.098277 seconds, 2304 images
```

Figura 17. Resumen del uso de GPU para 64 imágenes en batch.

Resultados para el caso Inclusión de técnica de AI para la catalogación y detección de rostros de personalidades de la cultura cubana.

En el campo de estudio de la presente investigación (la inteligencia artificial y el aprendizaje automático), de acuerdo a la literatura consultada [94]–[96], resulta útil incorporar para el análisis de los resultados la llamada matriz de confusión ya que esta se presenta como una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado. La matriz de confusión - *Error Matrix* no es más que una tabla que describe el rendimiento de un modelo supervisado en los datos de prueba, donde se desconocen los verdaderos valores [97]. Se llama de ese modo, porque a partir del análisis de la misma se hace que sea fácil detectar dónde el sistema está confundiendo dos clases o que tipos de aciertos y errores está teniendo el modelo a la hora de pasar por el proceso de aprendizaje con los datos. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real.

Tras un tercer entrenamiento los resultados fueron del todo positivos. Entiéndase que para esta ocasión los valores de las métricas *Acurracy, Recall* y *F1Score* fueron para los 100 rostros del modelo uno a entrenar. Para este propósito y en aras de ilustrar el contenido se escoge la matriz de confusión presentada. Nótese que no se elaboró una *Confussion Matrix* con las 100 clases pues resultaría imposible discernir la eficacia de la clasificación para los Verdaderos Positivos (del inglés *True Positive*, TP). En consecuencia y a partir de este momento para ilustrar y validar la propuesta se confeccionó la *Error Matrix* para pocas clases (10 clases). Nótese que cada numero de los mostrados en fondo azul denota la cantidad de imágenes detectadas para cada artista TP.

En busca de la data perfecta como reflejan varios autores, tras tres iteraciones realizadas se obtuvieron los mejores resultados. Para aquellas imágenes que no se identificó correctamente la persona se constató que las mismas tenían poses dorsales, expresiones atípicas, iluminaciones y porciones del rostro ocluidas (usaban gafas de sol), esta situación constituye una polémica demostrada en los trabajo de [98]–[101] y [44] para datas imperfectas que necesitan técnicas de detección que difieren de las utilizadas en la presente investigación.

Analizando los resultados obtenidos se puede apreciar que son muy similares a los estudiados durante la revisión bibliográfica que se llevó a cabo para la conformación de este proyecto. Tal es el caso de [93] que muestra cómo se desempeña un clasificador SVM, entrenado con 20 fotos de tres personas, para la detección de rostros en un video de youtube, evidenciándose valores de precisión entre un 83 y un 95% cuando los personajes se encuentran de frente a cámara durante la corrida del video y no en posiciones de perfil que dificultan la extracción de características del rostro. Está además el caso de [102] que entrena una CNN usando *Transfer Learning* tal y como se hizo en la presente investigación, con la diferencia de

TELEMÁTICA
REVISTA DIGITAL de las Tecnologías de la Información y las Telecomunicaciones
Revista Telemática. Vol.20 No.2 , Abril-Junio, 2021, p.43- 61 ISSN 1729-3804

que la usa para la detección de mascotas, en este caso de perros, partiendo de un conjunto de prueba del cual el algoritmo obtiene una precisión de un 80%. En [103] haciendo uso de la biblioteca de visión artificial OpenCV y de la biblioteca LIBSVM para llevar a cabo el modelo SVM realizaron una comparación entre 3 bases de datos BioID, CMU-MIT, y otra creada por el autor conformada por 200 rostros candidatos como máximo, y evaluaron los resultados en cuanto a la tasa de acierto y los falsos positivos, obteniéndose 97.2%, 83.1% y 93.5% de tasa de acierto, respectivamente. Por otro lado, la tesis [104] atraviesa por las mismas 3 primeras fases y algoritmos de detección facial empleados por este proyecto con la diferencia de que en la última fase para comparar y clasificar opta por la distancia euclidiana; OpenCV solicitará a la base de datos de personas los vectores almacenados (que contienen las 128 dimensiones extraídas de cada rostro) y por último OpenCV realiza una comparación de distancias entre los vectores de rostros del video y los vectores de la base de datos de personas; de este modo se deriva el reconocimiento facial de este proceso con precisiones entre 85% y 95% resultados que comprenden un rango muy similar a los datos obtenidos por esta investigación. La comparación con otras investigaciones para evaluar desempeño queda fuera de esta investigación ya que la base de dato fue creada para este proyecto exclusivamente y el mismo tiene como finalidad la comparación intrínseca de clasificadores y detectores usados, las métricas refridas y la representación de los resultados a través de la *Error Matrix*.

True Level		Alain Daniel	Alain Pérez	David Torres	Diana Fuentes	Diván	Edith Massola
	Alain Daniel	7					
	Alain Pérez		7				
	David Torres			7			
	Diana Fuentes				9		
	Diván					7	
	Edith Massola						6

Figura 18. Matriz de confusión obtenida para modelo entrenado con mejores resultados.

Partiendo de que el objetivo final para la detección de rostros parte de todo un proceso de reconocimiento facial "escalonado" que fue descrito previamente, se puede acotar que desde el punto de vista del uso de técnicas de comparación estas pueden utilizarse indistintamente tanto para el proceso de entrenamiento como para el proceso de clasificación. Para la realización de la comparativa entre las técnicas elegidas para el análisis en esta investigación de manera puntual se puede decir que, en este caso tanto el uso de KNN [105], como SVM [76] fueron técnicas para las cuales se emplearon las misma técnicas de detección de rostro, mientras que para *Deep Learning* se emplearon redes pre entrenadas CNN [18], lo que significó de gran ayuda para el trabajo. Vale la pena aclarar que todas las pruebas se realizaron con el mismo *data set* con el objetivo de realizar una comparación justa del problema y ver realmente el desempeño de cada una de las técnicas de manera independiente a partir de sus métricas de desempeño.

Acotar que para la construcción de los modelos clasificadores se utilizaron dos carpetas, una para el entrenamiento, con cuatro fotos al menos de 100 artistas cubanos diferentes, y otra carpeta para la validación del modelo la cual contenía una foto al menos de los 100 artistas elegidos, pero dichas fotos diferentes a las empleadas durante el entrenamiento. Cada imagen cuenta con una resolución típica de 225x225 y para cada artista se cuenta con imágenes que tienen poses diferentes, edades diferentes, diferentes expresiones, con/sin espejuelos, disímiles posiciones del rostro. De ellas son 38 mujeres y 62 hombres para un total de 754 imágenes (621 para *train* y 133 para *test*). Un análisis específico en este sentido, para el caso con las técnicas anteriormente mencionadas para la detección de rostro con las técnicas lineales se traduce en la existencia de un total de 13 imágenes (2,09%) con el uso de KNN y 10 imágenes

(1,6%) para SVM en las que no se pudieron detectar el rostro del artista. A juicio del investigador, este resultado habría incidido negativamente únicamente cuando el cúmulo de imágenes no detectadas eficientemente (13 o 10 imágenes) correspondiese únicamente a un solo artista, ya que no se tendrían características propias del mismo. Paralelamente, un mismo análisis empleando la red pre entrenada solamente no detectó rostro únicamente en cuatro imágenes (0.664%), mostrando una ligera superioridad en relación a los métodos discutidos anteriormente. Se debe destacar que el conjunto de imágenes no detectadas con *Deep learning* coinciden con el 30.7% del conjunto de las 13 imágenes no detectadas anteriormente, por lo que puede mostrar una mejoría en relación a la detección exitosa de los rostros a partir de las métricas devueltas y de la inspección visual de la correcta catalogación. El enfoque de [46] demuestra la incidencia que tiene en el por ciento de efectividad obtenido la técnica de extracción de características implementada, la base de datos y los patrones de clasificación situación que se tomó en consideración.

En contraste y analizando otro objetivo, el proceso empleando este algoritmo (*Deep learning*) muestra en contra el tiempo necesario para entrenarlo y extraer las características de todas las imágenes, tomando un total de tres horas en total mientras que por la variante anterior solamente tarda 4 minutos (para el *hardware* usado de simulación). Por ende el factor tiempo resulta una variable muy importante a considerar para cierto tipo de aplicaciones y es muy notoria la diferencia (tres horas vs. cuatro minutos). Cabe decir además, aunque fue un análisis que no se llevó a cabo que los resultados de ambas corridas tanto usando CPU o GPU no deberían de dar los mismo resultados debido al propio funcionamiento de una red neuronal convolucional, que posee aleatoriedad en sus iteraciones, lo que sí sería muy representativo son las corridas.

Desde el punto de vista de las métricas que definen la eficiencia del algoritmo se pueden observar en la siguiente tabla los resultados obtenidos tras el análisis. Nótese como para corroborar el primer análisis efectuado en función de la objetividad y eficacia de *Deep Learnin*g los valores obtenidos para cada una de las métricas representadas son ligeramente superiores. Estos resultados son semejantes a los obtenidos por [106] donde en función de las métricas KNN obtiene un mejor desempeño en la catalogación eficiente de rostros en relación a SVM. A tono con los resultados de [107] que denotan como el uso de CNN para la detección y SVM como clasificador ofrece mayor exactitud debido al efecto de clasificación en datas no lineales.

Tabla 2: Resultados de las métricas para las técnicas KNN, SVM y *Deep Learning*.

Técnica/Métrica	***Precision***	***Recall***	***f1-score***
HOG+KNN	0.93	0.91	0.9198
HOG+SVM	0.89	0.90	0.8949
CNN+SVM	0.95	0.92	0.9347

6. CONCLUSIONES

El presente trabajo aborda el procesamiento de las señales digitales de televisión aplicando técnicas de visión computacional para la detección de patrones de interés. La utilización de Yolo como un *transferlearnig* permitió la reutilización de redes previamente entrenadas, lo que disminuye el tiempo y la carga computacional, acelerando la detección de la configuración óptima de pesos de las redes neuronales entrenadas. El entrenamiento de nuevos objetos en la red Darknet resultó satisfactorio tras la identificación exitosa de 10 clases de objetos adicionales. La efectividad de la red reentrenada fue validada experimentalmente mediante el análisis de los resultados de la métrica mAP, donde se obtenienen valores semejantes a los referidos en la literatura, y una mejora de 0.01% tras la adecuación del *dataset.* El modelo fue implentado para su ejecución en nodos con GPU, reduciendo significativamente el tiempo de ejecución de las corridas realizadas. Para la detección y clasificación de rostros se utilizaron métricas de evaluación como son la precisión, la sensibilidad, el *f1-score* y la exactitud que mostraron la efectividad del modelo. El desempeño de la detección y el reconocimiento de rostros se ve afectado por la oclusión, problema que puede ser atenuado mediante la implementación de una red analítica convolucional para la detección, combinada con SVM para la clasificación. Las mejoras obtenidas son de 0.06%, 0.02% y 0.03% para las métricas de *Presicion*, *Recall* y *f1-score*, respectivamente.

Revista Telemática. Vol.20 No.2 , Abril-Junio, 2021, p.43- 61 ISSN 1729-3804

REFERENCIAS

[1] A. Camarillo, J. Ríos, y K.-D. Althoff, "Product Lifecycle Management as Data Repository for Manufacturing Problem Solving", *Mater. Basel Switz.*, vol. 11, n.º 8, ago. 2018, doi: 10.3390/ma11081469.
[2] R. Basir *et al.*, "Fog Computing Enabling Industrial Internet of Things: State-of-the-Art and Research Challenges", *Sensors*, vol. 19, n.º 21, p. 4807, 2019.
[3] L. Rouhiainen, "Inteligencia artificial", *Madr. Alienta Editor.*, 2018.
[4] A. Berlanga, "El camino desde la inteligencia artificial al Big Data", *Rev. Índice*, n.º 68, 2016.
[5] "Artificial intelligence systems forprogramme production and exchange", p. 22.
[6] "YOLO: Real-Time Object Detection". https://pjreddie.com/darknet/yolo/ (accedido ago. 30, 2020).
[7] "How to Implement a YOLO (v3) Object Detector from Scratch in PyTorch: Part 1", *KDnuggets*. https://www.kdnuggets.com/how-to-implement-a-yolo-v3-object-detector-from-scratch-in-pytorch-part-1.html/ (accedido ago. 03, 2020).
[8] Na8, "Detección de Objetos con Python", *Aprende Machine Learning*, jun. 24, 2020. https://www.aprendemachinelearning.com/deteccion-de-objetos-con-python-yolo-keras-tutorial/ (accedido jul. 29, 2020).
[9] "python - Cómo crear recuadros delimitadores alrededor de los ROI con TensorFlow". https://stackoverrun.com/es/q/11131674 (accedido jul. 29, 2020).
[10] J. Hosang, R. Benenson, y B. Schiele, "Learning non-maximum suppression", en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4507–4515.
[11] A. Kamal, "YOLO, YOLOv2 and YOLOv3: All You want to know", *Medium*, abr. 26, 2020. https://medium.com/@amrokamal_47691/yolo-yolov2-and-yolov3-all-you-want-to-know-7e3e92dc4899 (accedido jul. 29, 2020).
[12] R. Huang, J. Pedoeem, y C. Chen, "YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers", en *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 2503–2510.
[13] C. Gutiérrez Lancho, "Detección de armas en vídeos mediante técnicas de Deep Learning", 2019, Accedido: jul. 27, 2020. [En línea]. Disponible en: https://academica-e.unavarra.es/xmlui/handle/2454/33697.
[14] T.-Y. Lin *et al.*, "Microsoft COCO: Common Objects in Context", *ArXiv14050312 Cs*, feb. 2015, Accedido: jul. 27, 2020. [En línea]. Disponible en: http://arxiv.org/abs/1405.0312.
[15] A. Krizhevsky, I. Sutskever, y G. E. Hinton, "Imagenet classification with deep convolutional neural networks", en *Advances in neural information processing systems*, 2012, pp. 1097–1105.
[16] K. Simonyan y A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *ArXiv Prepr. ArXiv14091556*, 2014.
[17] F. N. Iandola, A. Shen, P. Gao, y K. K. Deeplogo, "Hitting logo recognition with the deep neural network hammer", *ArXiv Prepr. ArXiv151002131*, 2015.
[18] K.-W. Li, S.-Y. Chen, S. Su, D.-J. Duh, H. Zhang, y S. Li, "Logo detection with extendibility and discrimination", *Multimed. Tools Appl.*, vol. 72, n.º 2, pp. 1285–1310, 2014.
[19] X. Liang, S. Liu, Y. Wei, L. Liu, L. Lin, y S. Yan, "Towards computational baby learning: A weakly-supervised approach for object detection", en *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 999–1007.
[20] A. Shrivastava, A. Gupta, y R. Girshick, "Training region-based object detectors with online hard example mining", en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 761–769.
[21] X. Chen y A. Gupta, "Webly supervised learning of convolutional networks", en *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1431–1439.
[22] "FlickrLogos". https://www.uni-augsburg.de/en/fakultaet/fai/informatik/prof/mmc/research/datensatze/flickrlogos/ (accedido jul. 27, 2020).
[23] S. Romberg, L. G. Pueyo, R. Lienhart, y R. Van Zwol, "Scalable logo recognition in real-world images", en *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, 2011, pp. 1–8.
[24] Á. Casado García, "Guiando la creación de modelos de detección de objetos basados en deep learning", Universidad de la Rioja, 2018.

[25] darrenl, *tzutalin/labelImg*. 2020.
[26] A. Torres Alonso, "Detección de frutas en árboles", 2020.
[27] R. Khandelwal, "Implementing YOLO on a custom dataset", *Medium*, nov. 23, 2019. https://towardsdatascience.com/implementing-yolo-on-a-custom-dataset-20101473ce53 (accedido jul. 27, 2020).
[28] "Packages included in Anaconda 5.2.0 for 64-bit Windows with Python 3.6 — Anaconda documentation". https://docs.anaconda.com/anaconda/packages/old-pkg-lists/5.2.0/py3.6_win-64/ (accedido ago. 12, 2020).
[29] E. O. Kirkegaard y J. D. Bjerrekær, "The OKCupid dataset: A very large public dataset of dating site users", *Open Differ. Psychol.*, vol. 46, pp. 1–10, 2016.
[30] Á. Fernández Arce, "Reconocimiento biométrico facial basado en nuevas arquitecturas de aprendizaje profundo", B.S. thesis, 2018.
[31] J. R. G. Porras, J. M. B. Bermejo, y S. de Hematología, "Diagnóstico y tratamiento trombocitopatías".
[32] F. Alpizar, C. A. Harvey, M. Saborío-Rodríguez, B. Viguera, M. R. Martínez-Rodríguez, y R. Vignola, *Household survey of climate change perception and adaptation strategies of smallholder coffee and basic grain farmers in Central America 2004-2014. UK Data Service, Colchester, Essex, GBR*. UK Data Service Colchester, Essex, GBR., 2019.
[33] I. Jiménez Silva, "Reconocimiento facial basado en redes neuronales convolucionales", 2018.
[34] P. Viola y M. J. Jones, "Robust real-time face detection", *Int. J. Comput. Vis.*, vol. 57, n.º 2, pp. 137–154, 2004.
[35] F. Moreno y Y. Sisco, "Minería de datos para la detección de rostros mediante el uso de Maquinas de Soporte Vectorial, Redes Neuronales Artificiales y Redes Neuronales Convolucionales."
[36] "HOG Histograma de Gradientes Orientados", dic. 04, 2017. https://carlosjuliopardoblog.wordpress.com/2017/12/04/hog-histograma-de-gradientes-orientados/ (accedido ago. 13, 2020).
[37] I. del Burgo Ullate, "Detección automática de pólipos basada en el Histograma de Gradientes Orientados", 2017.
[38] E. O. Arroyave, "Detección automatizada de objetos en secuencias de video utilizando histogramas de gradientes orientados", PhD Thesis, Universidad Tecnológica de Pereira. Facultad de Ingenierías Eléctrica …, 2015.
[39] D. Catalán Vitas, "Detección automática de personas mediante Histograma de Gradientes Orientados", 2017.
[40] J. J. M. Carrillo, "Histograma orientado a gradientes con máquina de soporte vectorial, en la clasificación del alfabeto dactilológico", *Rev. Tecnológica-ESPOL*, vol. 30, n.º 2, 2017.
[41] V. Kazemi y J. Sullivan, "One millisecond face alignment with an ensemble of regression trees", en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1867–1874.
[42] T. Y. Arias, A. H. Cutipa, N. A. Ale, y M. C. Niquin, "DruBot: Prototipo robótico para autenticación por comparación de proporciones faciales para el control de asistencia y detectar la suplantación en evaluaciones", *TECNIA*, vol. 29, n.º 1, 2019.
[43] "Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning | by Adam Geitgey | Medium". https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78 (accedido ago. 12, 2020).
[44] "python — ¿Cómo escribir una matriz de confusión en Python?" https://www.it-swarm.dev/es/python/como-escribir-una-matriz-de-confusion-en-python/968371247/ (accedido ago. 14, 2020).
[45] "sklearn.metrics.confusion_matrix — scikit-learn 0.23.2 documentation". https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html (accedido ago. 14, 2020).
[46] "Matrix factorization (recommender systems)", *Wikipedia*. jun. 14, 2020, Accedido: jul. 08, 2020. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=Matrix_factorization_(recommender_systems)&oldid=962469899.
[47] B. Peláez Fernández, "Reconocimiento de caras en entornos no controlados", 2012.
[48] S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way", *Medium*, dic. 17, 2018. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (accedido sep. 03, 2020).

SOBRE LOS AUTORES

Jorge Armando Portal Díaz https://orcid.org/0000-0003-1360-4930, graduado de Ingeniería en Control Automático en el 2017 en la Universidad Central "Marta Abreu" de Las Villas, perteneciente al claustro de profesores del Departamento de Control Automático, profesor instructor del colectivo de Programación. Trabaja en la Dirección de Informatización y se desempeña como Especialista principal de Seguridad informática y SysAdmin del Cluster de Big Data. Ha participado en proyectos en varios proyectos Internacionales como el VLIR, el proyecto HPC Cuba y otros proyectos nacionales no asociados a programa.

Irina Siles Siles https://orcid.org/0000-0003-1360-4930, graduada de Ingeniería en Telecomunicaciones y Electrónica en el 2007 en la Universidad Central "Marta Abreu" de Las Villas, perteneciente al claustro de profesores del Departamento de Telecomunicaciones, profesor auxiliar del colectivo de Radiocomunicaciones, Máster en Telemática (UCLV, 2016). Sus intereses de investigación se centran en: codificación de fuente y canal, asignación dinámica del espectro, televisión digital, nueva generación de redes de comunicaciones.

Ania María Sánchez Pérez https://orcid.org/0000-0003-2393-9590, graduada de Ingeniería en Telecomunicaciones y Electrónica en el 2020 en la Universidad Central "Marta Abreu" de Las Villas.

Yusnavi Cárdenas Leiva https://orcid.org/0000-0002-7896-7663, graduada de Ingeniería en Telecomunicaciones y Electrónica en el 2020 en la Universidad Central "Marta Abreu" de Las Villas.

CONFLICTO DE INTERESES

No existen algún conflicto de intereses de los autores o de las instituciones a las cuales pertenece en relación al contenido del artículo aquí reflejado.

CONTRIBUCIONES DE LOS AUTORES

- Jorge Armando Portal Díaz: Diseño de los experimentos, la realización de los experimentos, el análisis de los datos, la realización de simulaciones por ordenador, revisión crítica de cada una de las versiones del borrador del artículo y aprobación de la versión final a publicar.
- Irina Siles Siles: Conceptualización del estudio, análisis de los datos, diseño de los experimentos, la extracción de resultados y conclusiones, preparación, creación, organización y desarrollo del artículo, correo corresponsal.
- Ania María Sánchez Pérez: Contribución a la idea, organización del artículo, el análisis de los datos, la preparación de las figuras, revisión crítica de cada una de las versiones del borrador del artículo y aprobación de la versión final a publicar
- Yusnavi Cárdenas Leiva: Contribución a la idea, el análisis de los datos, revisión crítica de cada una de las versiones del borrador del artículo y aprobación de la versión final a publicar.

Esta revista provee acceso libre inmediato a su contenido bajo el principio de hacer disponible gratuitamente investigación al público. Los contenidos de la revista se distribuyen bajo una licencia Creative Commons Attribution-NonCommercial 4.0 Unported License. Se permite la copia y distribución de sus manuscritos por cualquier medio, siempre que mantenga el reconocimiento de sus autores y no se haga uso comercial de las obras.